VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zobrazování vlasů a srsti metodou sledování paprsků

Raytracing Hair and Fur

2015

Bc. Roman Mátl

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Roman Mátl**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Zobrazování vlasů a srsti metodou sledování paprsků
Ray Tracing Hair and Fur

Zásady pro vypracování:

Cílem práce je detailně popsat a následně naimplementovat některé z technik pro ray tracing vlasů a chlupů. V práce popište aktuální algoritmy a metody používané v dané oblasti a vyberte alespoň jeden přístup, který následně detailně popište a naimplementujte. Součástí implementace algoritmu bude ukázková aplikace, která bude vhodným způsobem demonstrovat funkčnost a vizuální kvalitu algoritmu. Implementaci proveďte v jazyce C++ a OpenGL 4.x při dodržení Google C++ Style Guide a kód důkladně zdokumentujte pomocí nástroje Doxygen.

1. Seznamte se s aktuálními metodami vizualizace srsti a vlasů,
2. alespoň jednu vybranou metodu detailně popište a naimplementujte,
3. vygenerujte sadu testovacích modelů a zhodnoťte vizuální kvalitu dosažených výsledků.

Seznam doporučené odborné literatury:

[1] Qin H. et al. Cone Tracing for Furry Object Rendering. IEEE Transactions on Visualization and Computer Graphics. 2013.
[2] Nakamaru K. et al. Distance Aware Ray Tracing for Curves. 2012.
[3] Runz M. Real-Time Hair Simulation and Rendering. 2012.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Tomáš Fabián**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 7. května 2015 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2015 .....

Děkuji vedoucímu práce Ing. Tomáši Fabiánovi za odbornou pomoc a podnětné připomínky při psaní diplomové práce.

Abstrakt

Tato práce se zabývá problematikou zobrazování vlasů a srsti metodou sledování paprsků. Na začátku této práce je chronologicky zachycen vývoj algoritmů v dané oblasti. Dále jsou popsány tři aktuální techniky řešící tuto problematiku. Následuje detailní rozbor jedné z těchto aktuálních metod, která je schopna určit průsečík paprsku s křivkou bez nutnosti její aproximace pomocí polygonů, kdy není jasné, jak určit vhodnou míru tesselace. Tento přístup je výhodný, protože je zde vždy zaručeno hladké zobrazení vláken. Druhá část práce je zaměřena na konkrétní implementaci vybrané metody a vyhodnocení dosažených výsledků.

Klíčová slova: sledování paprsků, Béziérovy křivky, vlasy, srst, vlákna, hierarchie obalových těles

Abstract

This thesis deals with ray-tracing hair and fur. At the beginning of this thesis, the development of algorithms in given area is chronologically captured. The following describes the three current techniques addressing this issue. A detailed analysis of one of these current methods follows. The chosen method is able to determine the ray-curve intersection without using its approximation by polygons where it is not clear how to determine an appropriate tessellation rate. This approach is advantageous because it always ensures a smooth fibers rendering. The second part of the text focuses on the implementation of selected method and evaluation of results.

Keywords: ray tracing, Bezier curves, hair, fur, fibers, bounding volume hierarchy

Seznam použitých zkratek a symbolů

AABB	–	Axis Aligned Boudning Box
BFSDF	–	Bidirectional Fiber Scattering Distribution Function
BRDF	–	Bidirectional Reflectance Distribution Function
BVH	–	Bounding Volume Hierarchy
CPU	–	Central Processing Unit
DOF	–	Depth of Field
GLM	–	OpenGL Mathematics
MRT	–	Micropolygon Raytracing
OBB	–	Oriented Bounding Box
OpenCV	–	Open Source Computer Vision
OpenGL	–	Open Graphics Library
OpenMP	–	Open Multi-Processing
SRT	–	Stochastic Raytracing
SS	–	Supersampling

Obsah

Seznam obrázků

1 Úvod

Vlasy a srst jsou často nejdůležitější součástí většiny digitálně vytvořených postav, které jsou v této době prakticky nedílnou součástí filmového průmyslu, kde je kladen velký důraz na jejich realistickou kvalitu i za cenu delšího zpracování. Fotorealistické zobrazení vlasů a srsti bylo dlouhou dobu velkou výzvou pro nesčetné řady grafiků. Tento úkol je poměrně obtížný vzhledem k celé řadě komplexních grafických efektů. I přes značný vývoj v této oblasti je tento proces stále velmi časově náročný zvláště při efektech jako jsou například odraz a lom světla v kombinaci s kamerovými efekty jako například rozostření nebo rozmazání objektů při pohybu. Hlavní výzvou při renderování vlasů je tenká a velice početná geometrie jednotlivých vláken, která vyžaduje extrémně vysokou míru nad-vzorkování k vyprodukování kvalitního obrazu. Všechny tyto záležitosti dělají ze zobrazování vlasů a srsti metodou sledování paprsků velice zajímavou oblast, která je vhodná k dalšímu zkoumání a pátrání po co nejlepším řešení.

Cílem této diplomové práce je nastudovat a popsat vybrané aktuální algoritmy pro zobrazování vlasů a srsti metodou sledování paprsků. Dále z této oblasti zvolit jeden algoritmus a ten následně detailně rozebrat. Na jeho základě vytvořit ukázkovou aplikaci s implementací onoho algoritmu a pomocí vhodně vytvořené sady testovacích modelů demonstrovat funkčnost a vizuální kvalitu. Implementace bude provedena v programovacím jazyce C++.

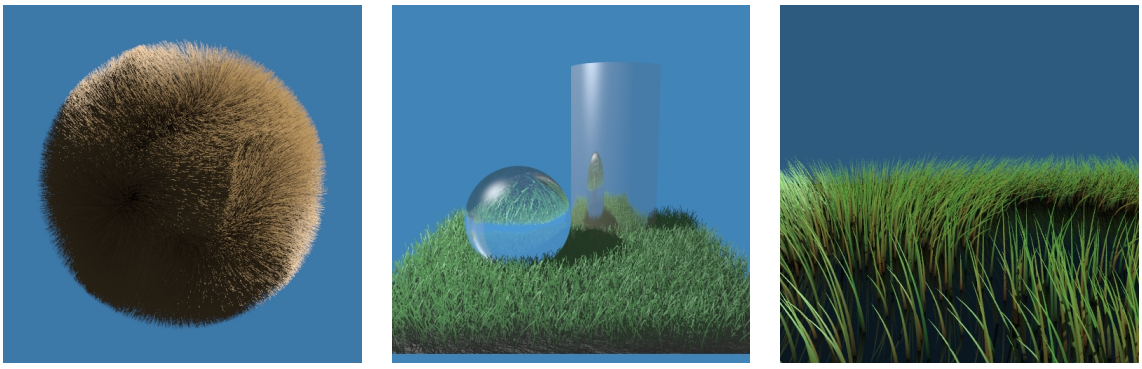
Text práce je rozdělen do sedmi oddílů. Po úvodu následuje kapitola, ve které je chronologicky popsána historie a vývoj metod, které nějakým způsobem ovlivnily aktuální moderní techniky v dané oblasti. V následující části jsou konkrétněji popsány tři vybrané aktuální přístupy. Ve čtvrté kapitole je velice detailně rozebrán zvolený algoritmus určený pro následovnou implementaci. Kapitola Implementace demonstrační aplikace se orientuje na popis realizace zvoleného algoritmu. Popisuje reprezentaci vlasů a srsti, výpočet kolizních testů paprsků a vláken, osvětlovací model, tvorbu testovacích modelů a další implementační detaily. Následuje kapitola věnována prezentaci dosažených výsledků a vyhodnocení jejich vizuální kvality.

2 Přehled používaných metod

V této kapitole jsou stručně popsány a chronologicky seřazeny vybrané metody nebo přístupy, které nějakým způsobem přispěly k vývoji v oblasti zobrazování vlasů a srsti metodou sledování paprsků. Jsou zde také zmíněny metody a přístupy, které byly publikovány do roku 2010 včetně. Na základech některých z těchto technik leží aktuální přístupy k problematice, které jsou blíže popsány v následující 3. kapitole.

V roce 1989 byl vydán článek s názvem Rendering fur with three dimensional textures [9, 10], ve kterém autoři J. T. Kajiya a T. L. Kay uvádějí způsob vykreslování scény s velice jemnými detaily v kombinaci s anizotropním osvětlením modelu, což zahrnuje i zobrazení vláken pomocí objektu nazvaného texel. Texel je renderovací geometrické primitivum inspirované voxely (částice objemu představující hodnotu v pravidelné mřížce třídimenzionálního prostoru). Vlákna vlasů jsou zde reprezentovány jako nekonečně tenké válce.

V roce 2002 byl vydán článek s názvem Ray tracing for curves primitive [14] od autorů Koji Nakamaru a Yoshio Ohno popisující jednoduchý framework. Tento obecný rámec slouží jako předpis pro kolizní testy mezi paprskem a křivkou pro metodu sledování paprsků. Avšak při zobrazování velice jemné geometrie vláken se algoritmus potýkal s problémy se vzorkováním a aliasingem. Cena k vyprodukování obrazu bez aliasingu byla značně vysoká.



(a) (b) (c)

Obrázek 1: Obrázky vytvořené metodou Ray tracing for curves primitive, efekty odrazu a lomu světla a stíny. Model na obrázku (a) je složen z 100 000 křivek (renderovací doba na CPU 18,9 minut). Na obrázku (b) je zobrazeno 10 000 křivek (renderovací doba na CPU 10,3 minut). Poslední obrázek je složen z 5 000 křivek (renderovací doba na CPU 7,5 minut). Zdroj: [14]

S. R. Marschner, H. W. Jensen, M. Cammarano, S. Worley a P. Hanrahan roku 2003 vydali článek s názvem Light Scattering from Human Hair Fibers [11, 12], ve kterém vyvinuli nový model pro řešení rozptylu světla pro jednotlivé vlasové vlákna. Tento model řešení předvádí vizuální efekty, které nebyly podchyceny modelem pánů Kajiya a Kay [9, 10]. Předchozí model byl rozšířen o simulaci mimo-rovinného rozptylu záření v kombinaci s vícenásobným spekulárním osvětlením a s variací rozptylu záření s rotací

kolem osy vlákna. Kvalita tohoto modelu je velice blízká realitě (viz porovnání na obrázku 2).

(a) (b) (c)

Obrázek 2: Porovnání modelu autorů Kajiya a Kay (a) a modelem navrženým autory Marschner a kol. (b). Na posledním obrázku (c) je fotografie reálné scény při stejném osvětlení. Zdroj: [11]

Článek z roku 2006 autora Maurice van Swaaij [23] popisuje metodu pro zobrazování srsti, peří a trávy pro potřeby tvůrců filmu Ice Age: Meltdown. Pro zajištění možnosti zobrazování vláken s efekty odrazu a lomu světla s efektem sebezastínění (self-shadowing), zde byla použita technika předzpracování geometrie vláken do tří dimenzionální voxelové sítě. Tímto přístupem je zajištěno omezení alokované paměti, protože každý voxel obsahuje souhrn geometrie, která se vyskytuje v jeho prostoru. Dále je aplikací filtrovací funkce zajištěn anti-aliasing v rámci každého voxelu. Ukázalo se, že procházení pravidelné voxelové sítě je v tomto případě rychlejší než při použití hierarchické akcelerační struktury u osově zarovnaných obalových těles, které se při jemné geometrii často překrývali. Nevýhodou tohoto přístupu je jeho nevhodnost pro efekt pohybového rozmazání, kterou autor řešil jinými technikami. Výsledek této práce je možné pozorovat na obrázku 3.

V roce 2007 byl autory Arno Zinke a Andreas Weber publikován článek [28] popisující nový jednotný formalismus pro model rozptylu světla od vláken z pohledu záření. Byl nazván Bidirectional Fiber Scattering Distribution Function (zkráceně BFSDF). V tomto článku bylo předvedeno, že je možné vidět konkrétní předchozí specializované přístupy pro zobrazování vlasů jako instance BFSDF. Kvůli častému výskytu vzdálených zdrojů světla a pozorovatelů byla odvozena efektivní aproximace tohoto komplexního modelu. Tento přístup poskytuje pevný základ pro srovnávání a klasifikaci vláken podle jejich vlastnosti rozptylu světla. Na obrázcích 4 je možné pozorovat výsledky aplikace BFSDF.

V roce 2007 byl publikován článek s názvem Stochastic Simplification of Aggregate Detail [4, 5], který reagoval na potřebu snížení paměťových a časových nároků potřebných pro renderování detailních scén s velkým počtem menších objektů. Za tímto účelem byla vyvinuta stochastická technika, při které jsou scény renderovány tak, že je náhodně

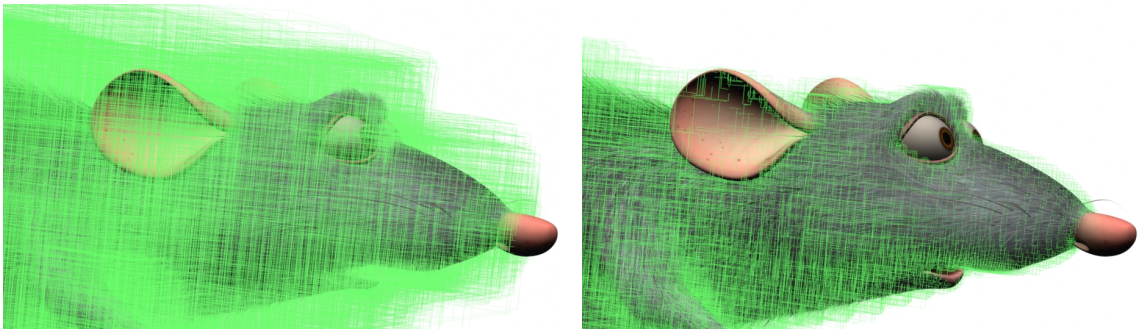
Obrázek 3: Obrázek scény s modelem zvířete s více jak 1,3 milióny vláken srsti vytvořeným pomocí technik popsaných v článku [23]

Obrázek 4: Různé výsledky aplikace BFSDF. Zdroj: [28]

vybrána podmnožina geometrických elementů (objektů). Ty jsou statisticky měněny tak, aby byl zachován celkový vzhled scény. Množství zjednodušení, kvalita a rychlost algoritmu závisí na mnoha faktorech jako například velikost obrazu, pohybovém rozmazání a rozostření.

Technika Stochastic Simplification of Aggregate Detail byla klíčová pro vytvoření známého filmu Ratatouille. Pro potřeby tvůrců tohoto animovaného filmu bylo zapotřebí hned několika dalších nových technologií představených téhož roku v technické zprávě 500 Million and Counting: Hair Rendering on Ratatouille [21]. Mezi tyto technologie patří Perfect Bounding, což je dvou průchodová metoda k vytvoření přesných obalových těles. Na obrázku 5 je možné vidět rozdíl mezi starším způsobem tvoření obalových těles a Perfect Boundingem.

V článku Interactive Hair Rendering Under Environment Lighting [19, 20] publikova-



(a) (b)

Obrázek 5: Rozdíl mezi starším způsobem tvorby obalových těles (a) a Perfect Boundingem (b). Zdroj: [21]

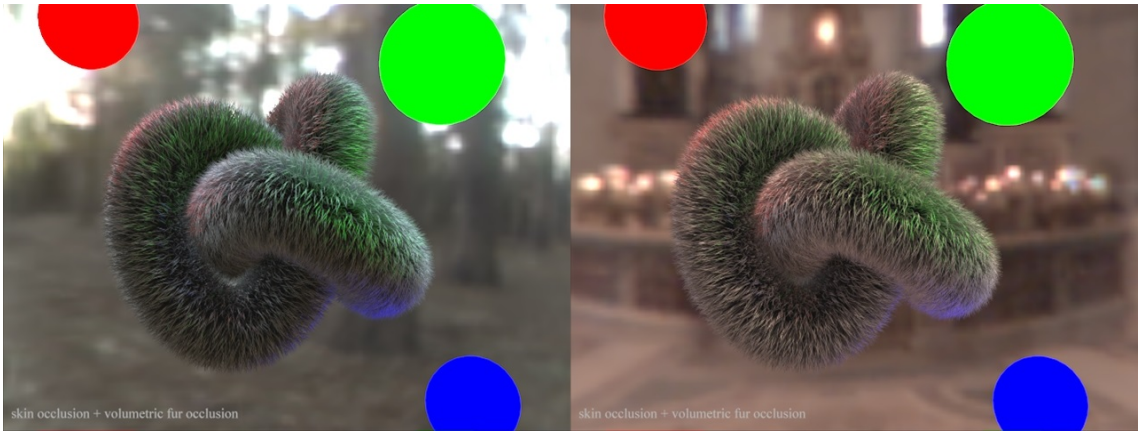
ném v roce 2010 byl představen algoritmus pro interaktivní zobrazování vlasů s jednoduchým i vícenásobným rozptylem světla za komplexního osvětlení prostředí. Pro účinný ray tracing zde byla vlákna vlasů aproximovány voxely. Na obrázku 6 je možné pozorovat výsledky tohoto algoritmu.

(a) (b)

Obrázek 6: Obrázky vytvořeny algoritmem popsaném v článku Interactive hair rendering under environment lighting, obrázek (a) je renderován s efektem jednoduchého rozptylu světla a na obrázku (b) s vícenásobným rozptylem světla. Model vlasů se skládá z 10 000 vláken a z 270 000 úseček. Zdroj: [19, 20]

Článek Fast Furry Ray Gathering [15] publikovaný v roce 2010 popisuje několik technik pro efektivní shromažďování difuzní a spekulární složky paprsků odražených od vláken vlasů. Byl vytvořen takzvaný cone-shell BRDF model (kuželem obalená obousměrná distribuční funkce odrazu světla), který je založen na reprezentaci vláken pomocí nekonečných válců z článku [9]. Nový model je obecnější, použitelný a realistický, přesto

zůstává relativně jednoduchý a rychlý pro výpočet. Tento model byl obohacen o důsledné testování okluze (překrytí) geometrie kůže. Na závěr byla k vylepšení výkonu nasazena technika mezipaměti stínování pro každý pramen vláken spojená s filtrovací technikou obdobnou trilineárnímu mip-mappingu. Na obrázku 7 je možné pozorovat výsledky algoritmu.



Obrázek 7: Obrázky s rozlišením 640×480 pixelů renderované algoritmem Fast Furry Ray Gathering. Scéna se skládá z 52 tisíc pramenů vláken s 50ti difuznímu a 30ti spekulárními shromažďovacími paprsky na stínovaný bod. Renderování jednoho obrázku trvalo 57 sekund (Athlon64 X2 2800MHz, jedno vlákno). Zdroj: [15]

3 Vybrané aktuální přístupy

V této kapitole je blíže popsáno několik vybraných relativně aktuálních metod pro zobrazování vlasů a srsti metodou sledování paprsků. U každé metody je popsán princip fungování, na jakou formu geometrie je přístup aplikován, jaké fyzikální vlastnosti světla řeší, posouzení vizuálních výsledků jednotlivých metod a popis výhod a nevýhod dané metody. Tyto metody jsem vybral hlavně proto, že jsou podle mého názoru hlavními představiteli třech různých směrů ke kterým ray tracing vlasů a srsti prozatím dospěl. Těmito směry jsou Clone tracing (místo sledování jednotlivých paprsků jsou sledovány svazky paprsků ve tvaru kuželu), tradiční technika (rozšíření základního ray tracingu pro geometrické primitivum trojúhelníku o objekt reprezentující vlákno) a posledním směrem, u kterého jsou všechny objekty scény složeny z velkého množství mikro-polygonů (tedy i vlákna), přičemž algoritmus musí být vysoce časově optimalizovaný.

3.1 Cone Tracing for Furry Object Rendering

Algoritmus podle stejnojmenného článku [18] poskytuje vysoce kvalitní a efektivní způsob zobrazování chlupatých objektů s relativně nízkou časovou náročností. Tento algoritmus je založen na metodě sledování paprsků, kde je místo tradiční reprezentace paprsku jako přímky použit kužel. Algoritmus pracuje s konkrétním typem geometrického primitiva scény, a to s vlákny reprezentovanými jako množinu propojených úseček s lineárně interpolovanými šířkami pro každý vrchol. Tato metoda řeší jev odrazu a lomu světla i efekt rozostřovaní pomocí agregace mnoha vzorkovacích paprsků ve formě jednoho kužele pro jednotlivé pixely výsledného obrazu, čímž značně redukuje vysoký počet paprsků pro supersampling (nad-vzorkování) potřebný k redukci aliasingu způsobeného velice tenkou geometrií vláken vlasů a srsti.

K redukci počtu kolizních testů vláken s kužely paprsků a k zajištění rychlejšího nalezení vláken, které se potenciálně protínají s kužely reprezentující paprsky je použita hierarchie obalových těles pro geometrii vláken. Algoritmus dále využívá množiny propojených pásek, které jsou aproximací projekce vláken do obrazové roviny. Výpočetní náročnost kompozice a filtrování průhledných vzorků je efektivně redukována aproximačním výpočtem variant stínování, zastínění a překrytí povrchů uvnitř každého kužele. Na obrázku 8 je možné pozorovat výsledky použití algoritmu Cone tracing for furry object rendering. Při renderování obrazu bylo dosaženo velice hladkých výsledků dokonce už při použití pouze jednoho vzorku pro každý pixel. I přes to, že časová náročnost pro výpočet jednoho vzorku je větší, než u tradičních metod, je tento přístup rychlejší pro komplexní scény s více detaily.

Hlavní výhodou algoritmu je, že znatelně redukuje čas potřebný k vytvoření realistického obrazu bez jakéhokoliv šumu díky nižším nárokům na počet vzorků pro supersampling. Metoda také podporuje efekty rozostření, pohybové rozmazání, odraz a lom světla při zobrazování vlasů a srstí. Nespornou výhodou je jeho jednoduchá integrace do existujících raytracing frameworků. Nevýhodou algoritmu je jeho nevhodnost pro zobrazování hustých pramenů srsti. Algoritmus se potýká s mírně rozmazanými měkkými stíny a odrazy světla. Další nevýhodou je, že při protínání většího množství pramenů srsti může

(a) Metoda 3.1, 87,8 s (b) SRT 4×4, 76,2 s (c) SRT 21×21, 513,5 s (d) SRT 31×31, 1224,3 s (e) MRT 21×21, 1053,1 s

Obrázek 8: Model veverky renderovaný s efektem hloubky ostrosti a s odrazem světla. Obrázek (a) je vytvořen metodou Cone Tracing for Furry Object Rendering. Obrázek (b) je vytvořen metodou stochastic ray tracing s 4×4 supersamplingem, který trvalo vytvořit přibližně stejnou dobu, ale očividně trpí artefakty šumu. Tento jev je patrný dokud není supersampling zvýšen na 21×21 vzorků na paprsek (obrázek (c)). Obrázek (d) je renderován při 31×31 supersamplingu. Podobný počet vzorků je potřeba i při použití metody Micropolygon raytracing [6] k vyprodukování obrazu srovnatelné kvality (obrázek (e)). Model je složen z 129 000 vláken rozdělených do 931 000 úseček. Zdroj: [18]

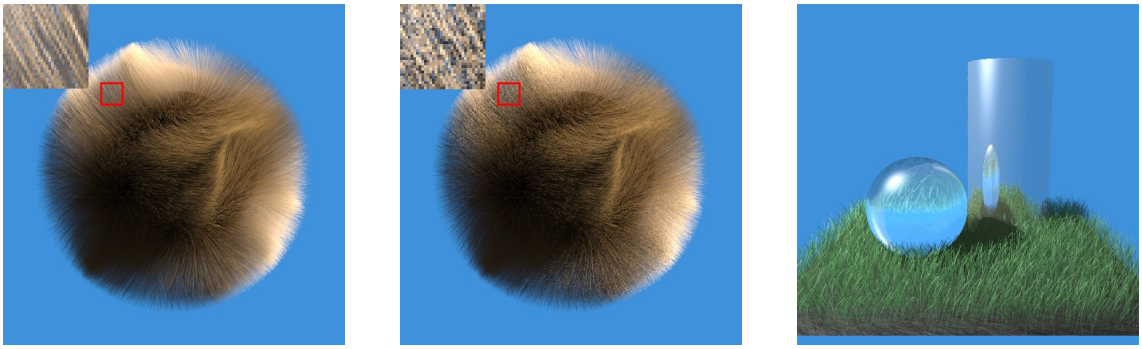
jeho rychlost výpočtu značně klesnout. Algoritmus neakceleruje renderování s efektem pohybového rozmazání [1].

3.2 Distance Aware Ray Tracing for Curves

Distance aware ray tracing for curves metoda je představena ve stejnojmenném článku [13]. Název „Distance-aware“ (v překladu „vědom si vzdálenosti“), je kvůli tomu, že tato metoda při výpočtu počítá nejen aktuální průsečíky (výsledky kolizních testů ray tracingu), ale i vzdálenosti ke křivkám. Algoritmus pracuje s reprezentací vláken ve formě křivek a využívá metodu diferenciál paprsku (ray differentials) popsánu v publikaci [8] k aproximaci překryvu křivek reprezentujících vlákna a k akumulaci jejich barev a zastínění ve správném pořadí.

Metoda rozšiřuje algoritmus kolizních testů mezi paprskem a křivkou tak, že není vypočten pouze skutečný průsečík, ale také „pseudo“ průsečík, tedy nejbližší bod ke křivce. Pomocí tohoto bodu a diferenciál paprsku v tomto bodě je aproximováno překrytí dané křivky. Dále je zde používána akcelerační struktura uniformní mřížky. Ta je spravována a procházena upravenou verzí algoritmu jménem slice-based packed traversal uvedenou ve článku [24, 25]. Tato akcelerační struktura umožňuje částečné seřazení průsečíků

[1] Zdroj: `http://www.zren.info/`.



(a) (b) (c)

Obrázek 9: Obrázek (a) je rendrován pomocí algoritmu 3.2 (1 vzorek na pixel, renderovací doba 6 minut), obrázek (b) je pro srovnání renderován tradičním přístupem (16×16 vzorků na pixel, renderovací doba 47 minut) a na obrázku (c) je scéna rendrována i s odrazem a lomem světla. Zdroj: [13]

od nejbližších po nejvzdálenější, ale bohužel není úplná garance toho, zdali bude seznam průsečíků opravdu seřazen. Kvůli časové náročnosti je tento seznam zcela seřazen pouze každý pátý krok při traverzaci (průchod) geometrie. Konečný příspěvek k celkové barvě každého průsečíku je řešen takzvanou akumulovanou průhledností, která je počítána už během kolizních testů. Pokud průhlednost překročí určitou prahovou hodnotu je výpočet traverzace pro daný paprsek ukončen. Tento přístup je použit i u odražených, lomených a stínových paprsků a je díky němu ušetřeno mnoho traverzačních kroků a kolizních testování.

Na obrázku 9 je možné pozorovat výsledky použití algoritmu Distance aware ray tracing for curves. Při renderování bylo dosaženo hladkých výsledků a to i dokonce při použití jednoho vzorku pro každý pixel. I přes to, že časová náročnost pro výpočet jednoho vzorku je větší než u tradičních metod, je tento přístup rychlejší pro komplexní scény s více detaily.

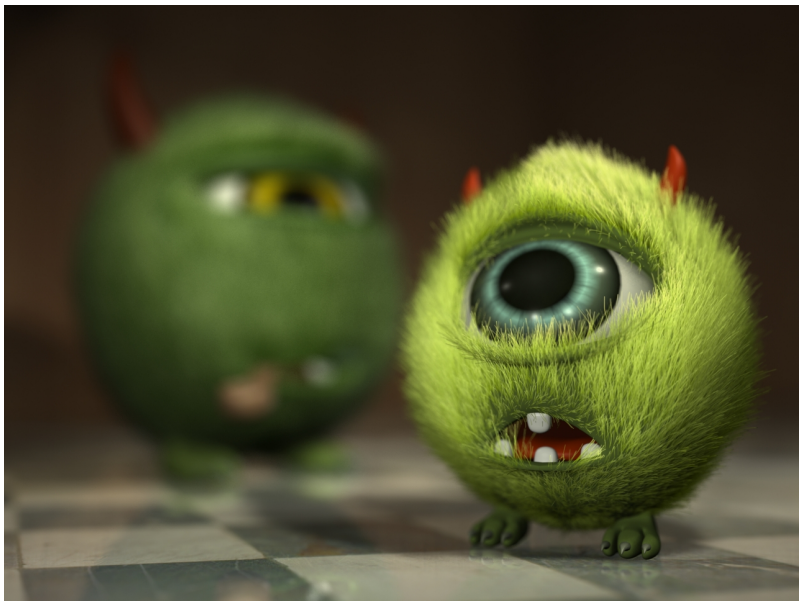
Velkou výhodou algoritmu je rychlost při renderování detailních scén. Naopak méně časově výhodné je jeho použití pro méně komplexní scény. Další jeho výhodou je, že nepotřebuje mnoho vzorků pro každý pixel k zabránění aliasingu.

3.3 Micropolygon Ray Tracing With Defocus and Motion Blur

Článek [6, 7] popisuje metodu založenou na principu micropolygon ray tracingu představeném v článku [2, 3] rozšířenou o podporu efektů rozostření (DOF) a pohybové rozmazání. Hlavním rozšířením z pohledu zobrazování vlasů a srsti je zavedení vylepšené hierarchie obalových těles (BVH) založené na čtyřrozměrných lichoběžnících, které jsou promítány do třírozměrných orientovaných obalových těles. Tato struktura je hned z několika důvodů velice vhodná pro trasování rozostřených a pohybem rozmazaných mikropolygonů. Prvním je využití čtvrté dimenze k zaznamenání času potřebného k efektu pohybového rozostření. Další výhodou je použití orientovaných obalových těles (OBB),

které v prostorových dimenzích zamezuje degeneraci traverzace pro rozsáhlé scény. Třetí výhodou je, že paměť potřebná pro BVH strukturu je přirozeně omezená. Hlavní výhodou je možnost efektivně vytvořit tuto strukturu využitím mikropolygonové sítě, která se vytváří již při teselaci geometrie na mikropolygony.

Algoritmus řeší průhlednost materiálů tak, že při kolizních testech zaznamená nejbližších m průsečíků. Počet zpracovávaných průhledných vrstev v jednom průchodu je omezen kvůli paměťové náročnosti. Po každém průchodu jsou všechny vytvořené vrstvy akumulovány do finálního obrazu a všechna paměť potřebná pro výpočet je uvolněna. Tento přístup se nazývá paměťově omezená průhlednost. Při vhodné volbě parametru m se zobrazování scén se srstí či vlasy výrazně akceleruje a to bez viditelných artefaktů výsledného obrazu. Na obrázku 10 je možné vidět vyrenderovanou scénu algoritmem Micropolygon Ray Tracing With Defocus and Motion Blur, kde bylo zapotřebí využít paměťově omezenou průhlednost. Počet průhledných vrstev byl omezen na pět.



Obrázek 10: Obrázek vytvořený algoritmem Micropolygon Ray Tracing With Defocus and Motion Blur. Scéna je renderována při rozlišení 1280×960 pixelů (13×13 supersampling) s efektem rozostření a odrazu světla a obsahuje 17,6 milionů mikropolygonů (14,3 mikropolygonů na jeden pixel). Renderovací čas byl 221 sekund. Zdroj: [6]

Hlavní výhodou algoritmu je jeho nezávislost na vstupní geometrii, která je vždy teselována na mikropolygony, tedy není zapotřebí mít speciální geometrickou reprezentaci pro vlákna vlasů či srsti a s tím spojené další speciální úpravy algoritmu. Další výhodou je velice skvělá kvalita výstupního obrazu a to zejména s efekty rozostření a pohybového rozmazání.

Nevýhodou algoritmu je, že je i přes velkou míru akcelerace relativně pomalejší, zejména při použití velice komplexních shaderů. I přesto je tento algoritmus velice zajímavým konkurentem aktuálních ray tracerů. Dalším omezením algoritmu je neefektivní

řešení průhlednosti. Tento algoritmus dokáže zpracovat větší množství průhledných vrstev, ale efektivita BVH znatelně klesá při trasování paprsků začínajících uvnitř objektů. Tím se značně prodlužuje doba zpracování.

Z těchto zajímavých a aktuálních metod byla pro další zkoumání vybrána metoda Distance aware ray tracing for curves 3.2. Důvodem je jednak existující dostatek materiálů pro její implementaci, ale i její vlastnost používání geometrie křivek jako reprezentaci vláken. Lze ji tedy použít jako rozšíření tradičního ray traceru. Tato metoda bude blíže rozebrána v následující kapitole.

4 Algoritmus pro zobrazování vlasů a srsti

Jak již bylo řečeno, pro další zkoumání byla vybrána metoda Distance aware ray tracing for curves 3.2. Tato metoda bude následně velice podrobně rozebrána. Nejdříve bude popsána datová reprezentace vláken (křivek), poté bude rozebrán základní kolizní test paprsku a křivky. Poté bude následovat popis výpočtu diferenciál paprsku a nakonec bude popsána tvorba a metoda pro průchod hierarchické akcelerační struktury, kterou autor metody používá. Na základě informací obsažených v této kapitole vytvořím implementaci vlastního ray traceru vlasů a srsti.

4.1 Reprezentace křivky

Algoritmus používá reprezentaci 3D křivek ve formě `Curves` primitiv definovaných v knize [1]. Toto primitivum může reprezentovat pásky (neboli stuhy, pro představu například reprezentace plochého stébla trávy) i vlákna. Každé `Curves` primitivum specifikuje jednu či více 3D pásek dané šířky popsané pomocí množiny řídících bodů. Křivky mohou být lineární nebo kubické. Šířka po celé délce křivky může být specifikována jako konstanta nebo jako proměnný parametr (interpolace parametru šířky mezi sousedními body křivky zadaných jako seznam šířek). Každá křivka vytváří plochou pásku, přičemž kontrolní body specifikují pouze směr osy („páteře“) křivky. Ve standardním případě se páska bude vždy natáčet tak, aby její normála byla kolineární s pohledovým vektorem kamery (bude se natáčet na kameru) a tímto způsobem simulovala úzký válec. Nicméně pokud jsou zde specifikovány normálové vektory po délce pásky, budou použity k jejímu řízení (páska se nebude implicitně natáčet na kameru). Díky tomuto přístupu je umožněna kontrola nad rotací pásky a lze jí využít například pro modelování stébel trávy.

4.2 Základní kolizní test paprsku a křivky

Základem kolizního testu paprsku a křivky je rámcová technika (framework), která je specifikována pro Béziérovi křivky a je popsána v článku autorů Koji Nakamaru a Yoshio Ohno [14]. Výhodou tohoto přístupu je jeho nezávislost na konkrétní reprezentaci křivky. Je ho také poměrně snadné přizpůsobit pro nový druh křivky. Tento algoritmus se skládá ze dvou částí, projekce a dělení prostoru. Nejprve je provedena projekce křivky do dvourozměrného (x, y) souřadného systému a to ortogonální projekcí ve směru paprsku. Po této projekci paprsek začíná v počátku souřadném systému a pokračuje podél osy z. Projekce je složena z translace a rotace definované následujícími transformačními maticemi:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -o_x & -o_y & -o_z & 1 \end{pmatrix} \begin{pmatrix} l_z/d & -l_x l_y/d & l_x & 0 \\ 0 & d & l_y & 0 \\ -l_x/d & -l_y l_z/d & l_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{1}$$

kde (l_x, l_y, l_z) je normalizovaný směrový vektor paprsku, (o_x, o_y, o_z) je počátečním bodem paprsku a $d = \sqrt{{l_x}^2, {l_z}^2}$. Matice rotace by měla být nahrazena maticí takovou, která rotuje $\pm\pi/2$ radiánu kolem osy x, pokud je d rovno nule. Směr rotace je určen znaménkem složky l_y. Smyslem této transformace je, že práce s křivkami v dvourozměrném prostoru redukuje výpočetní čas.

Následuje rekurzivní dělení křivky. Rekurze je omezena předem danou hloubkou zanoření z důvodů efektivity algoritmu a vyhnutí se výpočetně náročnému testu pro ověření, zdali interpolace části křivky úsečkou nevnese příliš velkou chybu ve výpočtu polohy průsečíku. Klíčovým aspektem je určení rozumné hloubky zanoření. Autor článku používá Wangovu metodu [26], podle které je určena hloubka zanoření r_0 pro danou maximální chybu vzdálenosti bodu interpolované křivky od odpovídajícího bodu na původní křivce (ϵ). Pro kontrolní body (x_i, y_i), kde $i \in 0, .., n$, je hodnota r_0 určena následovně:

$$L_0 = \max_{0 \leq i \leq n-2} (|x_i - 2x_{i+1} + x_{i+2}|, |y_i - 2y_{i+1} + y_{i+2}|) \,,$$

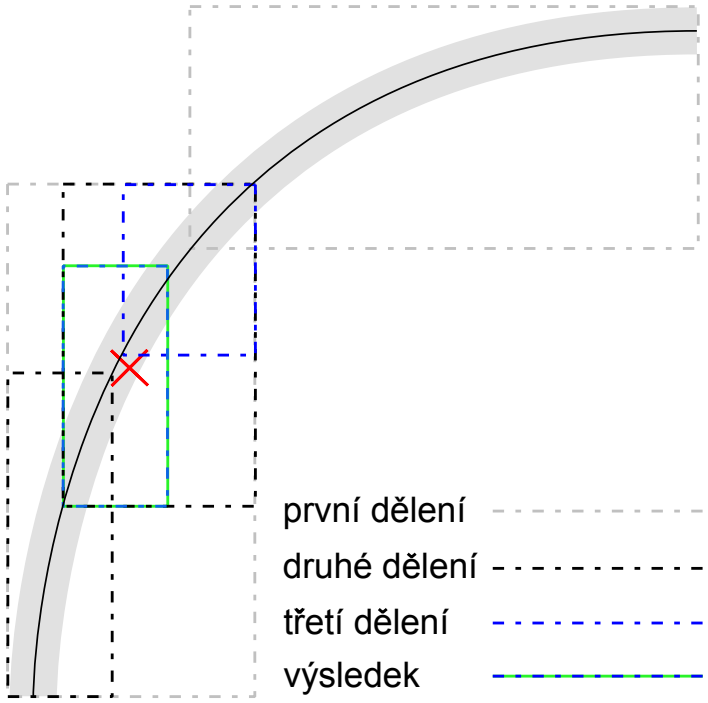
$$r_0 = \log_4 \frac{\sqrt{2} n(n-1) L_0}{8\epsilon} \,. \tag{2}$$

Hodnota parametru ϵ pro získání přijatelných výsledků byla empiricky určena jako jedna dvacetina šířky pásku.

V rámci rekurzivní procedury se nejdříve provede vytvoření obalového tělesa křivky (pásku) ve tvaru kvádru. Obalové těleso pro Bézierovu křivku lze udělat například tak, že je nejprve vytvořeno minimální obalové těleso pomocí prvního a posledního řídícího bodu, které je postupně rozšiřováno. Rozšiřování je provedeno na základě výpočtů extrémů křivky (pomocí parciálních derivací podle os x, y, z). Adekvátní rozšíření obalového tělesa je provedeno, pokud nalezený extrém leží vně aktuálního obalového tělesa.

Dále je proveden test, zdali obalové těleso křivky překrývá počátek souřadného systému, tedy bod $(0, 0)$. Aby nastala tato situace musí obalové těleso křivky překrývat malý čtverec se středem v bodě $(0, 0)$. Velikost stran onoho malého čtverce je dána jako maximální šířka pásku. V ose z probíhá porovnání hodnoty minimální z-ové souřadnice obalového tělesa s hodnotou vzdálenosti bodu pozorování po nejbližší již nalezený průsečík aktuálně zpracovávaným paprskem (pokud ještě nebyl nalezen žádný průsečík je tato hodnota nastavena jako nějaké velké číslo, ideálně největší číslo pro použitý datový typ). Na základě tohoto porovnání je zajištěno, aby byl nalezen průsečík, který je nejblíže k bodu pozorování. Poté je ještě provedeno porovnání hodnoty maximální z-ové souřadnice obalového tělesa s hodnotou konstanty nazvané `kEpsilon`, což je prahová konstanta určující minimální rozlišovací přesnost (typicky malá hodnota, například $0,0001$). Pomocí těchto testů jsou izolovány (odstraněny) segmenty křivky. Pokud obalové těleso segmentu křivky nepřekrývá malý čtverec se středem v bodě $(0, 0)$, tak pásek pro daný segment křivky nemůže překrývat bod $(0, 0)$ a tedy zde nemůže být nalezen žádný průsečík. Izolování segmentů křivky při rekurzivním dělení je znázorněno na obrázku 11.

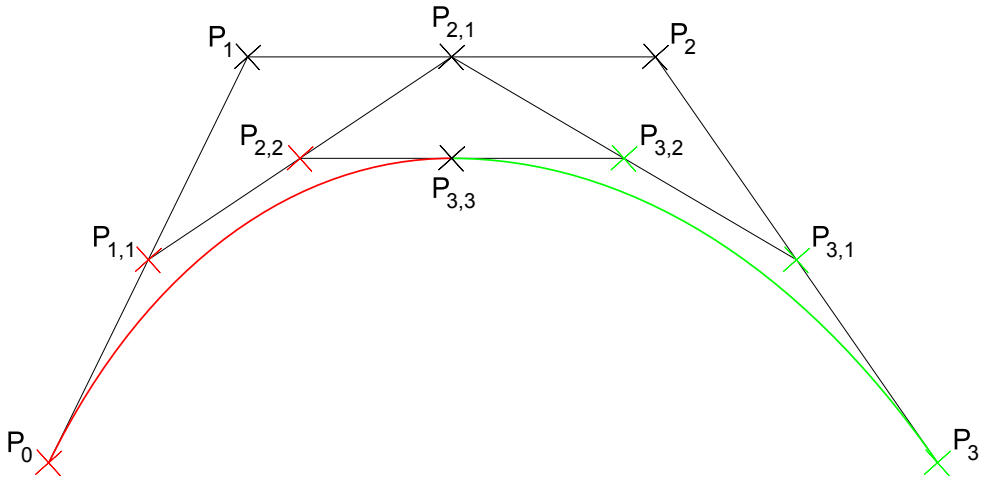
Pokud ještě nebylo dosaženo maximální hloubky rekurze je křivka rozdělena na poloviny a procedura je znovu rekurzivně volána pro oba fragmenty křivky. Na rozdělení Bézierovi křivky lze použít mírně modifikovaný algoritmus de Castaljau (blíže popsán



Obrázek 11: Demonstrace rekurzivního dělení křivky. Na demonstračním obrázku je zachycen pomocí různě barevných obálek segmentů původní křivky (vlákna) postup rekurzivního dělení křivky za účelem nalezení částí, ve kterých je potenciální možnost kolize paprsku s křivkou. Po dosažení maximální hloubky rekurze (v tomto případě třetího zanoření) algoritmus pracuje s výsledným segmentem křivky (zelenomodrý obal). Následný postup je zachycen na obrázku 13

dále v odstavci 4.2). Nejdříve jsou určeny hodnoty parametrů v_0 a v_n pro obě výsledné křivky podle parametrů v_0 a v_n původní křivky. Například pro původní křivku s parametry $v_0 = 0$ a $v_n = 0$ jsou pro první půlku křivky hodnoty $v_0 = 0$ a $v_n = 0,5$ a pro druhou půlku $v_0 = 0,5$ a $v_n = 1$ (viz obrázek 11). Obrázek 12 demonstruje dělení Bézierovi křivky třetího stupně na dvě poloviny pomocí algoritmu de Casteljau. Nejprve je proveden algoritmus de Castaljau pro určení bodu odpovídajícího parametru v_n první výsledné křivky, tedy bod který určuje střed původní křivky (bod $P_{3,3}$). Nově vzniklé pomocné body vytvořené algoritmem de Casteljau určují řídící polygony nových křivek. Pro první křivku jsou řídící body $P_0, P_{1,1}, P_{2,2}, P_{3,3}$ a pro druhou křivku $P_{3,3}, P_{3,2}, P_{3,1}, P_3$.

Jakmile je dosažena maximální hloubka rekurze je zpracovávaný fragment křivky interpolován úsečkou, která je určena prvním (x_0, y_0) a posledním (x_n, y_n) řídícím bodem fragmentu křivky. Dále je určena hodnota parametru w pro parametrické vyjádření bodu na interpolační úsečce, který zároveň leží na přímce, která je kolmá k interpolační úsečce



Obrázek 12: Dělení Bézierovi křivky třetího stupně na dvě poloviny (v bodě odpovídajícímu parametru $v = 0,5$). Řídící body nově vzniklých křivek (označeny barvou odpovídající barvě segmentů původní křivky) jsou určeny pomocí bodů vzniklých při provádění algoritmu de Casteljau za účelem zjištění polohy bodu odpovídajícímu parametru v. Postup vytvoření pomocných bodů je demonstrován na obrázku 14

a prochází bodem $(0, 0)$. Tento bod s bodem $(0, 0)$ určuje nejkratší vzdálenost mezi interpolovanou úsečkou a bodem $(0, 0)$. Na obrázku 13 je zobrazena ukázková situace při které je určován parametr w. Hodnota parametru w je vypočtena podle následujícího vztahu:

$$w = -\frac{x_0(x_n - x_0) + y_0(y_n - y_0}{(x_n - x_0)^2 + (y_n - y_0)^2}\,. \tag{3}$$

Pokud je jmenovatel tohoto zlomku roven nule, je úsečka přeskočena (neexistuje průsečík). Poté je hodnota parametru w ořezána na interval $\langle 0, 1\rangle$, tedy pokud bude hodnota větší než jedna bude nastavena na jedna a pokud bude hodnota menší než nula bude nastavena na nulu. Následně je na základě w určen parametr v, který je parametrem pro parametrické vyjádření konkrétního bodu v rámci celé původní křivky před rekurzivním dělením. Parametr v je vypočten podle následujícího vztahu:

$$v = v_0(1 - w) + v_n w\,, \tag{4}$$

kde v_0 a v_n odpovídají bodům (x_0, y_0) a (x_n, y_n). Po každém rekurzivním dělení křivky jsou předávány i patřičné hodnoty v_0 a v_n. Například při dělení původní křivky na dvě poloviny jsou pro první půlku křivky hodnoty $v_0 = 0$ a $v_n = 0,5$ a pro druhou půlku $v_0 = 0,5$ a $v_n = 1$ (viz obrázek 11). Tato lineární aproximace křivky může vnést do výpočtu chybu, ale tato chyba nikdy nebude díky předem dané hloubce rekurze větší než ϵ. Navíc

při zmenšování parametru ϵ se hloubka rekurze drasticky nezvyšuje a většina původní křivky je ořezána již při relativně malé hloubce rekurze. Proto je možné použít velice malou hodnotu ϵ bez významného ovlivnění výpočetního času.

Poloha bodu na Bézierově křivce n-tého stupně pro $n+1$ řídících bodů $(P_0, \ldots, P_n)$ pro daný parametr $v \in \langle 0, 1 \rangle$ je definována jako:

$$C(t) = \sum_{i=0}^{n} B_{i,n}(t) P_i \,, \tag{5}$$

přičemž $B_{i,n}(t)$ je i-tý Breinsteinův polynom n-tého stupně je dán vztahem:

$$B_{i,n} = \binom{n}{i} t^i (1-t)^{n-i} \,. \tag{6}$$

Pro výpočet bodu na Bézierově křivce lze také použít algoritmus de Casteljau, který spočívá v postupném dělení úseček řídícího polygonu (ten je určen množinou řídících bodů $P_0, \ldots, P_n$ pro Bézierovu křivku stupně n) v poměru daném parametrem $v \in \langle 0, 1 \rangle$. Počet nově vzniklých bodů se v každém kroku zmenšuje o jedničku. Ve chvíli, kdy zůstane jediný bod, je tento bod onen hledaný bod. Graficky znázorněný algoritmus de Casteljau je možné pozorovat na obrázku 14.

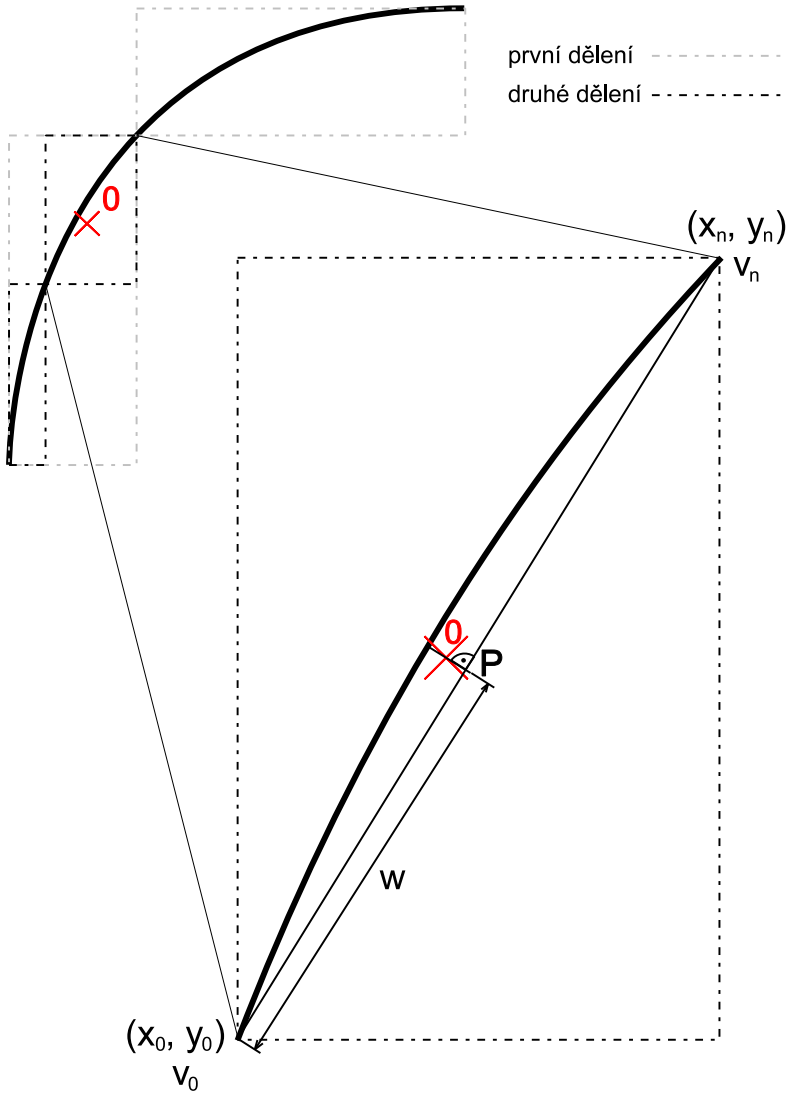
Pro potřeby algoritmu 3.2 je tento kolizní test dále rozšířen tak, aby navíc vracel „pseudo“ průsečík. Tento průsečík je nejbližší bod ke křivce, která zbyla po rekurzivním dělení a její pokrytí je aproximováno s ohledem na diferenciály paprsku pro daný bod. Výpočet diferenciál paprsků bude popsán v následující podkapitole 4.3. Diferenciály paprsku budou použity k odvození velikosti malého čtverce se středem v bodě $(0, 0)$ místo hodnoty šířky pásku.

4.3 Výpočet diferenciál paprsku

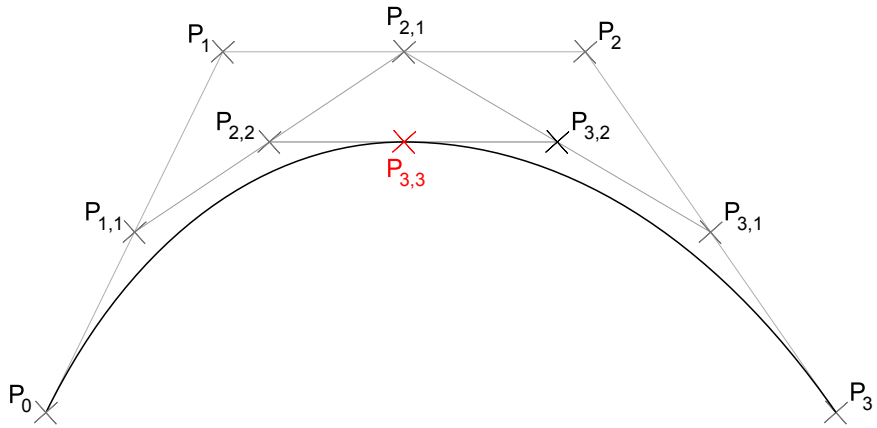
Výpočet a použití diferenciál paprsku je popsán v článku [8]. Diferenciály paprsků nesou informaci o vzdálenosti dopadu sousedních paprsků na povrch objektů při trasování konkrétního paprsku (lze si je představit jako „otisky stop“ sousedních paprsků). Tato informace může být použita k efektivnímu odstranění aliasingu bez nutnosti nad-vzorkování (supersamplingu) obrazové roviny. Cena výpočtu a sledování diferenciál paprsku je relativně malá vzhledem k ceně ostatních operací ray traceru.

Výpočet diferenciál paprsku je, jak již název napovídá, založen na diferenciálním kalkulu. Základní rovnice jsou odvozeny výpočet přenosu, odrazu a lomu světla (refrakci). Konkrétní rovnice jsou specifikovány na základě normál povrchů geometrie scény. V nadcházejících podkapitolách bude paprsek $\vec{R}$ (symbol šipky je zde použit pro reprezentaci vektoru, který je obohacen o souřadnici jeho počátku) reprezentován počátečním bodem (P) a normovaným směrovým vektorem (**d**):

$$\vec{R}(t) = P + \mathbf{d}t \,. \tag{7}$$



Obrázek 13: Interpolace křivky po dosažení maximální hloubky rekurze. Šedé čerchované obdélníky znázorňují první rekurzivní dělení křivky, černé čerchované obdélníky znázorňují druhé rekurzivní dělení. V tomto případě byla hloubka rekurze rovna dvěma. V detailu lze pozorovat informace potřebné pro výpočet parametrů w. Výsledný segment křivky je interpolován úsečkou tvořenou body X_0 a X_n. Bod $(0, 0)$ reprezentuje bod průsečíku paprsku s rovinou, na kterou je promítnuta křivka. Pomocí parametru w je určen bod P, který reprezentuje bod, který je nejblíže k bodu $(0, 0)$ a leží na interpolační úsečce. Všimněte si, že s větší hloubkou rekurzivního dělení křivky interpolační úsečka lépe odpovídá segmentu křivky



Obrázek 14: Určení bodu na Bézierově křivce třetího stupně odpovídající hodnotě parametru $v = 0,5$ pomocí algoritmu de Casteljau. Algoritmus nejdřív určí na úsečkách tvořených řídícími body bod P_x, který odpovídá opět parametru v tento pro parametrické vyjádření úsečky (příklad vztahu pro výpočet bodu $P_{1,1}$: $P_{1,1} = P_0 + v\mathbf{s}$, kde $\mathbf{s}$ je směrový vektor úsečky tvořené body P_0 a P_1.) Takto určené body ($P_{1,1}$, $P_{2,1}$, $P_{3,1}$) tvoří další úsečky, na kterých se opět hledá bod odpovídající parametru v. Tento postup se opakuje v závislosti na stupni Bézierovi křivky, obecně končí ve chvíli, kdy je vypočten jen jeden nový bod. V tomto případě končí určením bodu $P_{3,3}$, který je hledaným bodem

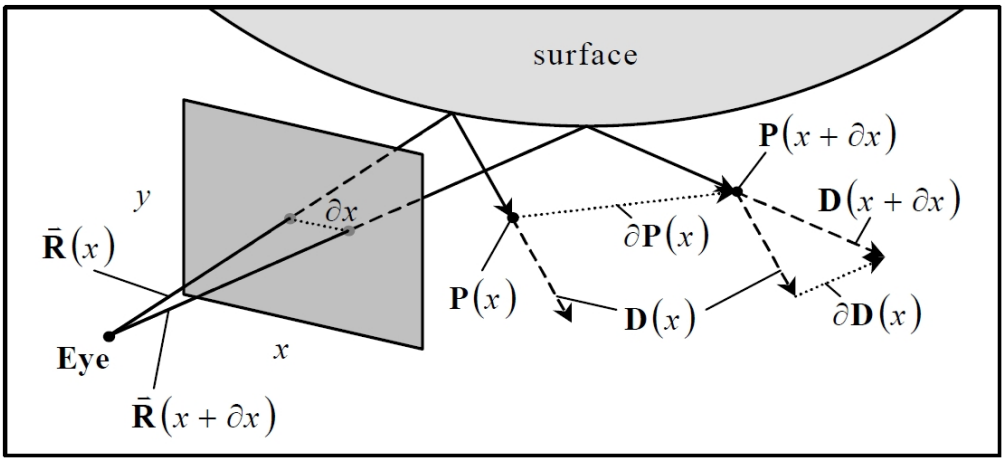
Počáteční hodnoty paprsku jsou závislé na parametrizaci obrazové roviny. Kamera je reprezentována bodem určujícím její polohu (Eye), směrem pozorování (**view**) a polohou zpracovávaného pixelu na obrazové rovině (x_{right}, y_{up}). Směr paprsku procházejícího pixelem obrazové roviny je dána lineární kombinací:

$$\mathbf{d}(x,y) = \mathbf{view} + \mathbf{x}_{\text{right}} + \mathbf{y}_{\text{up}}\,, \tag{8}$$

$$\begin{aligned} P(x,y) &= Eye\,, \\ \mathbf{d}(x,y) &= \frac{\mathbf{d}}{(\mathbf{d}\cdot\mathbf{d})^{\frac{1}{2}}}\,. \end{aligned} \tag{9}$$

Pro rozlišení horního a pravého sousedního paprsku je použita jejich reprezentace jako páru směrových vektorů, nazvaných diferenciály paprsku:

$$\begin{aligned} \frac{\partial\vec{R}}{\partial x} &= \left(\frac{\partial P}{\partial x}, \frac{\partial D}{\partial x}\right)\,, \\ \frac{\partial\vec{R}}{\partial y} &= \left(\frac{\partial P}{\partial y}, \frac{\partial D}{\partial y}\right)\,. \end{aligned} \tag{10}$$



Obrázek 15: Diferenciály paprsku. Schéma na obrázku ilustruje pozice a směry paprsku a odlišné odsazení paprsku po odrazu. Diferenciály paprsku reprezentují rozdíly mezi těmito pozicemi a jejich směry. Zdroj: [8]

Na obrázku 15 je znázorněn diferenciál paprsku. Výpočet diferenciálu paprsku v průběhu trasování paprsku scénou je možné provést zároveň s výpočtem nové pozice a směrového vektoru prozářeného nebo odraženého paprsku pomocí aproximace parciálních derivací prvního řádu pomocí následujících vztahů:

$$[\vec{R}(x+\Delta x,y)-\vec{R}(x,y)] \approx \Delta x \frac{\partial \vec{R}(x,y)}{\partial x}\,,$$

$$[\vec{R}(x,y+\Delta y)-\vec{R}(x,y)] \approx \Delta y \frac{\partial \vec{R}(x,y)}{\partial y}\,. \tag{11}$$

Nyní je možné vypočítat počáteční hodnotu diferenciálu paprsku ve směru osy x a to parciální derivací počátečního bodu paprsku a jeho směrnice (vztahy 9) podle souřadnice x:

$$\frac{\partial P}{\partial x} = 0\,,$$

$$\frac{\partial D}{\partial x} = \frac{(\mathbf{d}\cdot\mathbf{d})\mathbf{x}_{\text{right}} - (\mathbf{d}\cdot\mathbf{x}_{\text{right}})\mathbf{d}}{(\mathbf{d}\cdot\mathbf{d})^{\frac{3}{2}}}\,. \tag{12}$$

Vztahy pro výpočet diferenciál paprsku ve směru osy y mohou být odvozeny obdobným způsobem. Ačkoliv jsou sledovány pouze derivace prvního řádu, mohou být počítány derivace i vyšších řádů kvůli lepší aproximaci nebo za účelem omezení velikosti chyb při výpočtu. Avšak autorem algoritmu bylo zjištěno, že nespojitosti omezují efektivitu aproximací derivacemi vyšších řádů a že v praxi jsou pro dostatečně přesný výpočet postačující derivace prvního řádu. V následujících třech podkapitolách budou

odvozeny vztahy pro výpočet diferenciál paprsku pro tři běžné operace algoritmu sledování paprsků. Jedná se o přenos, reflexi a refrakci paprsku. Všechny výrazy jsou vyjádřeny obecně jen ve směru osy x, protože výpočet ve směrech ostatních os probíhá obdobně.

4.3.1 Přenos paprsku

Přenos je jednoduchá operace, při které se paprsek šíří skrz homogenní médium k bodu střetu paprsku s povrchem objektu scény (tedy k nejbližšímu nalezenému průsečíku). Vztah pro přenos paprsku na průsečík ve vzdálenosti t je dán:

$$\begin{aligned} P' &= P + t \cdot D\,, \\ D' &= D\,. \end{aligned} \tag{13}$$

Výpočet diferencí paprsku pomocí derivace vztahů 13 vypadá následovně:

$$\begin{aligned} \frac{\partial P'}{\partial x} &= \left(\frac{\partial P}{\partial x} + t\frac{\partial D}{\partial x}\right) + \frac{\partial t}{\partial x}D\,, \\ \frac{\partial D'}{\partial x} &= \frac{\partial D}{\partial x}\,. \end{aligned} \tag{14}$$

Pro rovinný povrch N (definovaný jako těžiště bodů P' tak, že $P' \cdot N = 0$) je parametr t dán vztahem:

$$t = -\frac{P \cdot N}{D \cdot N}\,. \tag{15}$$

Po derivaci a úpravě vztahu je výsledkem:

$$\frac{\partial t}{\partial x} = -\frac{\left(\frac{\partial P}{\partial x} + t\frac{\partial D}{\partial x}\right) \cdot N}{D \cdot N}\,. \tag{16}$$

Čtvrtá složka komponenty N je irelevantní v této rovnici, jelikož její skalární součin je počítán pouze ze složek určujících směr. Proto lze na komponentu N pohlížet jako na normálu povrchu. Rovnice 14 a 16 mají odpovídající geometrickou interpretaci. První rovnice vyjadřuje fakt, že se paprsek šíří homogenním prostředím. Rozdíl vzdáleností pozic derivovaného offsetu (odsazení) paprsku se mění v závislosti na směrovém offsetu paprsku a s jeho uraženou vzdáleností. Výraz 16 reprezentuje rovnoběžnou projekci těchto rozdílů vzdáleností pozic ve směru paprsku do roviny. Tento postup lze aplikovat i na obecný povrch (tedy ne jenom na rovinné povrchy), stačí pouze použít adekvátní normálu N povrchu v bodě průsečíku s paprskem.

4.3.2 Reflexe

Vzhledem k tomu, že byl paprsek přenesen na povrch objektu pomocí rovnic 13, jsou rovnice pro odražený paprsek na základě rovnic 11 definovány následovně:

$$\begin{aligned} P' &= P\,, \\ D' &= D - 2N(D \cdot N)\,. \end{aligned} \tag{17}$$

Diferenciál paprsku pro odražený paprsek je určen jako:

$$\begin{aligned} \frac{\partial P'}{\partial x} &= \frac{\partial P}{\partial x}\,, \\ \frac{\partial D'}{\partial x} &= \frac{\partial D}{\partial x} - 2\left[(D \cdot N)\frac{\partial N}{\partial x} + \frac{\partial (D \cdot N)}{\partial x}N\right]\,, \end{aligned} \tag{18}$$

kde:

$$\frac{\partial (D \cdot N)}{\partial x} = \frac{\partial D}{\partial x} \cdot N + D \cdot \frac{\partial N}{\partial x}\,. \tag{19}$$

Tyto rovnice vyžadují výpočet derivaci normály povrchu v průsečíku s paprskem. Tento výpočet je rozebrán v nadcházející podkapitole .

4.3.3 Refrakce

Jakmile je paprsek přenesen na povrch objektu, je možné definovat rovnici pro refrakci paprsku následovným způsobem:

$$\begin{aligned} P' &= P\,, \\ D' &= \eta D - \mu N\,, \end{aligned} \tag{20}$$

kde je použit zkrácený zápis:

$$\begin{aligned} \mu &= \left[\eta(D \cdot N) - (D' \cdot N)\right]\,, \\ D' \cdot N &= -\sqrt{1 - \eta^2\left[1 - (D \cdot N)^2\right]}\,, \end{aligned} \tag{21}$$

η je relativní index lomu pro přechod z prostředí s indexem lomu η_1 do prostředí s indexem lomu η_2 a je definován následovně:

$$\eta = \frac{\eta_2}{\eta_1}\,. \tag{22}$$

Následnou derivací rovnic 20 je získáno:

$$\frac{\partial P'}{\partial x} = \frac{\partial P}{\partial x}\,,$$
$$\frac{\partial D'}{\partial x} = \eta\frac{\partial D}{\partial x} - \left(\mu\frac{\partial N}{\partial x} + \frac{\partial \mu}{\partial x}N\right)\,, \tag{23}$$

kde lze parciální derivaci μ vyjádřit s ohledem na rovnici 19 následovně:

$$\frac{\partial \mu}{\partial x} = \left[\eta - \frac{\eta^2(D\cdot N)}{(D'\cdot N)}\right]\frac{\partial(D\cdot N)}{\partial x}\,. \tag{24}$$

4.3.4 Normály povrchů

Pro výpočet výrazů odvozených k určení diferenciál paprsku při reflexi a refrakci v podkapitolách 4.3.2 a 4.3.3 je potřeba znát první parciální derivaci normované normály podle osy x (pro ostatní osy výpočet bude opět vypadat obdobně). V diferenciální geometrii je „operátor tvaru“ S definován jako záporná derivace normovaného vektoru normály vzhledem ke směru tečny povrchu. Tento operátor zcela popisuje diferenciálně malou plochu na povrchu objektu. Výpočet směrnice tečny v bodě průsečíku paprsku s povrchem objektu je dán parciální derivací onoho průsečíku (P), tedy:

$$\frac{\partial N}{\partial x} = -S\left(\frac{\partial P}{\partial x}\right) \tag{25}$$

Pro rovinný povrch je operátor tvaru roven nule. V případě kulovitého povrchu je operátor tvaru dán normovaným vektorem určujícím tečnu povrchu, která je zmenšena o obrácenou hodnotu poloměru koule určující sférické zakřivení onoho povrchu. Vzorce pro výpočet operátorů tvaru parametrických i implicitních ploch je možné nalézt v knize o diferenciální geometrii [22].

Výpočet diferenciál paprsku při průchodu scénou je využit nejen pro lepší antialiazing při texturování objektů. Ale i pro potřeby zobrazování vlasů a srsti. Zde je důležitý pro výpočet pseudo-průsečíků, které jsou zapotřebí k aproximaci překrytí křivky. V následující podkapitole bude popsána hierarchická akcelerační struktura použitá autorem algoritmu.

4.4 Hierarchická akcelerační struktura

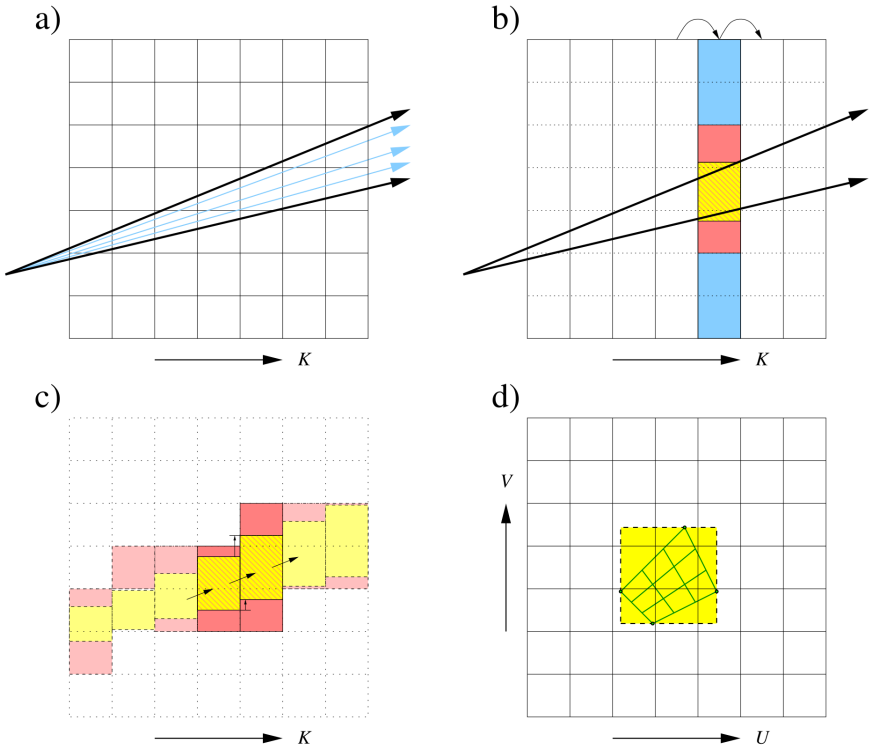
Scény zobrazované metodou sledování paprsků jsou často složeny z obrovského množství geometrických primitiv. Testování všech těchto primitiv, jestli existuje průsečík se všemi sledovanými paprsky, je časově příliš náročné. Proto je vhodné tento proces zefektivnit, a to nejlépe pomocí nějaké vhodné hierarchické akcelerační struktury. Pro účely zvoleného algoritmu zobrazování vlasů a srsti byla jeho autorem zvolena hierarchická akcelerační struktura ve formě uniformní sítě popsané v publikacích [25], [24]. Průchod touto sítí je

založen na metodě nazvané Slice-based Packet Traversal, která pro svazek několika sousedních paprsků určí průřez uniformní sítě a prověřuje jednotlivé buňky průřezu sítě od nejbližší k nejvzdálenější po jednotlivých vrstvách. Použití akcelerační struktury je velice podstatné kvůli redukci času zobrazování (v ideálním případě z původní polynomiální časové složitosti na logaritmickou složitost) s tím, že přibude potřebný čas na vytvoření této struktury v době inicializace. Avšak tento čas je téměř zanedbatelný v porovnání s časem potřebným k zobrazení komplexní scény. V následujícím textu bude podrobněji popsána tato akcelerační struktura a metoda pro její průchod.

4.4.1 Vytváření akcelerační struktury

Nejprve je zapotřebí vytvořit uniformní síť a patřičně roztřídit jednotlivá geometrická primitiva do adekvátních buněk sítě. Nejdříve se transformují paprsky do souřadného systému uniformní sítě, ve které je $N_x \times N_y \times N_z$ mapa buněk sítě v trojrozměrné oblasti $\langle 0..N_x) \times \langle 0..N_y) \times \langle 0..N_z)$. V tomto souřadném systému lze jednoduše vypočítat souřadnice příslušné buňky pro jakýkoliv bod v prostoru tak, že se ořežou desetinné části všech jeho souřadnic. Poté se vybere takzvaná dominantní komponenta (jedna z os $\pm X$, $\pm Y$ nebo $\pm Z$) směru prvního paprsku. Toto bude takzvaná hlavní osa průchodu (dále značená $\vec{K}$). Všechny další paprsky prochází strukturu podél této zvolené osy, přičemž zbylé dva rozměry jsou označeny $\vec{U}$ a $\vec{V}$. Všechny paprsky, které jsou součástí vyšetřovaného svazku musí mít ve směru hlavní osy průchodu stejný směr (znaménko), aby bylo možné procházet geometrická primitiva od nejbližších k nejvzdálenějším. K porušení tohoto požadavku dochází pokud je úhel tvořen směrovými vektory dvou paprsku větší než $\frac{\pi}{2}$ radiánů. Tento požadavek není potřeba ověřovat při vysílání primárních paprsků (paprsky s počátkem v bodě pozorování scény).

Nyní předpokládejme průřez (k) uniformní sítí ve směru hlavní osy průchodu $\vec{K}$. Pro každý paprsek r_i z vyšetřovaného svazku existuje bod p_i^{in}, ve kterém paprsek vstupuje do průřezu a bod p_i^{out}, ve kterém z něj vystupuje. Osově zarovnaný kvádr β, který obsahuje všechny tyto body (pro všechny paprsky ve svazku) pokrývá všechny buňky uniformní sítě v daném průřezu, které byly navštíveny alespoň jedním paprskem (tento kvádr je znázorněn červenou barvou na obrázku 16(b)). Jakmile je určen β, ořezáním desetinných částí jeho nejmenších a největších souřadnic jsou zjištěny u a v rozsahy všech buněk řezu k, do kterých zasahuje alespoň jeden paprsek ze svazku. Situace je znázorněna na obrázku 16(d).



Obrázek 16: Demonstrace algoritmu Slice-based Packet Traversal. Nejdříve je určena obálka svazku vyšetřovaných paprsků (a). Poté jsou (obrázek (b)) určovány kolize obálky s jednotlivými průřezy, čímž jsou určeny buňky uniformní sítě, kterými prochází alespoň jeden paprsek z onoho svazku (červená barva). Následně dojde k určení minimální obálky paprsků ve všech průřezech (oranžovou barvou na obrázcích (b) a (c)). Nakonec jsou obdobně zjištěny u a v rozsahy všech buněk řezu k, do kterých zasahuje alespoň jeden paprsek ze svazku (d). Zdroj: [24]

5 Implementace demonstrační aplikace

Vytvořil jsem konzolovou aplikaci za účelem vyzkoušení a experimentování se zvoleným algoritmem pro zobrazování vlasů a srsti metodou sledování paprsků. K implementaci aplikace jsem použil programovací jazyk C++ a knihovnu OpenCV. Pro některé matematické výpočty jsem použil knihovnu OpenGL Mathematics (GLM) a k paralelizaci výpočtu na CPU jsem použil OpenMP, což je soustava direktiv pro překladač a knihovních procedur pro paralelní programování.

V následujících částech kapitoly popíšu zvolenou strukturu pro reprezentaci vláken (křivek). Dále rozeberu konkrétní implementaci algoritmu pro kolizní testování paprsku a křivky. Poté následuje popis výpočtů potřebných informací pro vytvoření osvětlovacího modelu scény. Následně popíšu začlenění algoritmu do implementovaného zobrazovacího frameworku. Na konci kapitoly zmíním tvorbu procedurálně vytvářených modelů testovacích scén.

5.1 Reprezentace vláken

Nejprve bylo zapotřebí zvolit a na implementovat takovou reprezentaci vlasů a srsti, pro kterou bude zvolený způsob provádění kolizních testů (teorii je možné nalézt v kapitole 4.2) správně pracovat. V kapitole 4.1 jsem popsal reprezentaci vláken (křivek), kterou používájí autoři metody Distance Aware Ray Tracing for Curves 3.2. Touto reprezentací jsem se nechal inspirovat. Pro kolizní testy paprsku a křivky využili autoři metody přístupu popsaného v kapitole 4.2, který je jakýmsi frameworkem a pro reprezentaci křivek připouští jakoukoliv křivku, kterou lze popsat parametrickou rovnicí. Z tohoto důvodu jsem zvolil velice známý druh křivky a to Bézierovu křivku (podrobné informace je možné nalézt v knize [17]). Konkrétně jsem zvolil Bézierovu křivku třetího řádu a to hned z několika důvodů. Prvním důvodem je to, že počáteční a koncový bod křivky je totožný s počátečním koncovým řídícím bodem. Tato vlastnost je velice důležitá pro snadné napojování křivek na sebe a tedy i vytváření procedurálního modelu. Dalším důvodem je existence velice příhodného rekurzivního algoritmu zvaného de Casteljau pro výpočet bodu na Bézierově křivce. Tento algoritmus také využiji pro dělení křivek. Poměrně výhodnou vlastností je i to, že existuje možnost převodu téměř libovolného aktuálně používaného druhu křivky na Bézierovu křivku. Poslední příhodnou vlastností pro zvolený algoritmus na kolizní testování je, že se křivka nachází uvnitř konvexní obálky tvořené jejími řídícími body.

V pseudokódu 1 je možné vidět klíčové vlastnosti struktury reprezentující vlákno. Jde o reprezentaci Bézierovi křivky s tím, že je navíc přidána informace o šířce vlákna a také informace, zdali je nutno při výpočtu normály počítat s křivkou jako s vláknem nebo jako s páskou (k tomuto tématu se ještě vrátím v kapitole 5.3). Z důvodu redukce redundantních výpočtů je v každé křivce informace o projekční matici pro transformaci křivky do obrazové roviny kolmé na směrnici paprsku (tato informace je opět potřebná pro výpočet kolizních testů). Struktura dále nese informace o čtyřech kontrolních bodech křivky a kopii jejich původních souřadnic. Kopie je uchovávána z optimalizačních důvodů.

```
struct BezierCurve
{
  Vertex controlPoints[4]; /* čtyři kontrolni body, které se podle
      potřeby transformují */
  Vertex originalControlPoints[4]; /* kopie kontrolních bodů ze stavu
      při vytvoření křivky */
  float width; /* šířka vlákna (křivky) */
  bool ribbon; /* příznak, zda-li při výpočtu normály povrchu brát
      křivku jako vlákno nebo pásek */
  Matrix4x4 * ProjectionMatrix; /* projekční matice pro transformaci
     křivky do obrazové roviny pro výpočet kolizních testů */
  Model * model; /* Odkaz na model, který obsahuje odkazy
      na volumetrické textury, směrnicové mapy a další potřebné
      informace pro správný běh algoritmu */
}

struct Vertex
{
  Vector3 position; /* pozice vertexu */
  Vector3 normal; /* normála vertexu */
  Vector3 color; /* RGB barva vertexu (pokud nejsou specifikovány
      texturovací souřadnice) */
  Vector2 textureCoords; /* Texturovací souřadnice */
}
```

Pseudokód 1: Pseudokód popisující strukturu reprezentující vlákno (křivku) v demonstrační aplikaci. Vysvětlení a účel jednotlivých složek struktury je popsán pomocí komentářů vložených do pseudokódu

Struktura má odkaz na model, který obsahuje odkazy na volumetrické textury, směrnicové mapy a další potřebné informace pro správný běh algoritmu.

Chtěl bych se také podotknout, jaký má tato reprezentace vláken význam oproti reprezentaci vláken pomocí aproximace jejich povrchů trojúhelníky. Toto srovnání provedu na jednoduchém příkladě. Křivku je možné interpolovat množinou úseček. Aby průměrná křivka vypadala relativně přijatelně je zapotřebí minimálně x úseček. Tento odhad lze přenést na vlákno, které lze interpolovat množinou válců s osami odpovídajícími adekvátním interpolačním úsečkám. Nyní je zapotřebí triangulovat (interpolovat pomocí trojúhelníků) všechny válce opět takovým způsobem, aby se zobrazení přijatelně podobalo realitě. Výsledkem tedy bude y trojúhelníků pro každý válec. Dohromady lze tedy jedno vlákno reprezentovat c trojúhelníky (kde $c = xy$). Tento fakt by znamenal, že by kolizní test paprsku a trojúhelníku a případné určení normály jeho povrchu muselo být c krát rychlejší než kolizní test a případný výpočet normály křivky (vlákna). Pokud bych odhadl hodnoty těchto proměnných jako $x = 12$ a $y = 10$ (tento odhad je poměrně mírný a výsledná kvalita vlákna by byla poměrně nízká), tak by musel být kolizní test trojúhel-

níku sto dvacet krát rychlejší. Skutečnost je ovšem taková, že pokud jsou vlákna relativně malá nebo ve velké vzdálenosti od bodu pozorování, tak by bylo možné je interpolovat pouze jako stuhu, tedy výše zmíněné interpolační válce nahradit čtyřúhelníky. Přitom každý čtyřúhelník lze triangulovat pomocí dvou trojúhelníků. Tímto by se proměnná y redukovala na hodnotu dva, ale i tak by byl součin c při teoretickém odhadu relativně vysoký. Původní předpoklad ovšem byl, že mezi jednotlivými časovými hodnotami tak velký rozdíl nebude. Reálné porovnání těchto hodnot je možné sledovat v kapitole 6.

5.2 Kolizní test paprsku a křivky

Nejpodstatnější částí raytraceru jsou metody pro výpočet kolizních testů paprsků s geometrickými primitivy. Vzhledem k tématu této práce je jádrem programu právě kolizní test paprsku s křivkou. Implementaci tohoto kolizního testu jsem vytvořil na základě teorie popsané v kapitole 4.2. Nejprve popíši algoritmus pro výpočet kolizních testů z vyšší úrovně abstrakce a poté se zaměřím na detailnější části.

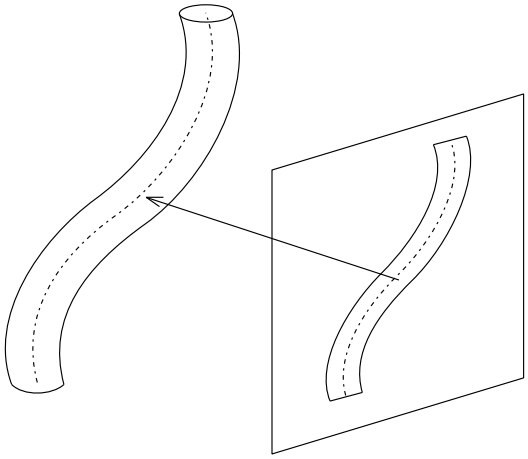
Algoritmus se skládá ze dvou hlavních částí, transformace a rekurzivní dělení křivky. Nejprve je provedena transformace křivky do dvourozměrného prostoru, kterou si lze představit jako rovnoběžnou projekci na rovinu, jejíž normála odpovídá směrnici vyšetřovaného paprsku. Bod, který by odpovídal průsečíku paprsku s touto rovinou je zároveň středem souřadného systému prostoru, ve kterém se nachází křivka po transformaci (na obrázku 17 se nachází nákres této situace). Projekce se provádí vynásobením všech řídících bodů křivky projekční maticí. Tuto matici je zapotřebí vypočítat pro každý vrhaný paprsek zvlášť. Projekční matice je určena jako součin dvou transformačních matic, konkrétně maticí translační a rotační. Předpis pro výpočet těchto dvou matic je možné nalézt v kapitole 4.2 (vzorec 1). Je také ještě je zapotřebí otestovat, zdali odmocnina součtu kvadrátu x-ové a z-ové souřadnice směrnice paprsku není rovna nule. V tomto případě je matice rotace nahrazena maticí reprezentující transformaci rotace o úhel $\pm\pi/2$ okolo osy x, kde znaménko rotace určuje znaménko y-ové souřadnice směrnice paprsku. Matice rotace pro kladné znaménko vypadá následovně:

$$\begin{pmatrix} \cos(\frac{\pi}{2}) & \sin(\frac{\pi}{2}) & 0 & 0 \\ -\sin(\frac{\pi}{2}) & \cos(\frac{\pi}{2}) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{26}$$

a rotace o úhel $-\frac{\pi}{2}$ kolem osy x je dána maticí:

$$\begin{pmatrix} \cos(-\frac{\pi}{2}) & \sin(-\frac{\pi}{2}) & 0 & 0 \\ -\sin(-\frac{\pi}{2}) & \cos(-\frac{\pi}{2}) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{27}$$

Po transformaci křivky následuje rekurzivní dělení křivky. Nejprve je nutné určit hloubku zanoření rekurze. Hloubka zanoření je počítána podle vztahů 2. Hodnota parametru ϵ rovna jedné dvacetině šířky pásku (vlákna) se ukázala jako dostačující pro dosažení přijatelných výsledků. Experimentoval jsem také s konstantní hodnotou hloubky



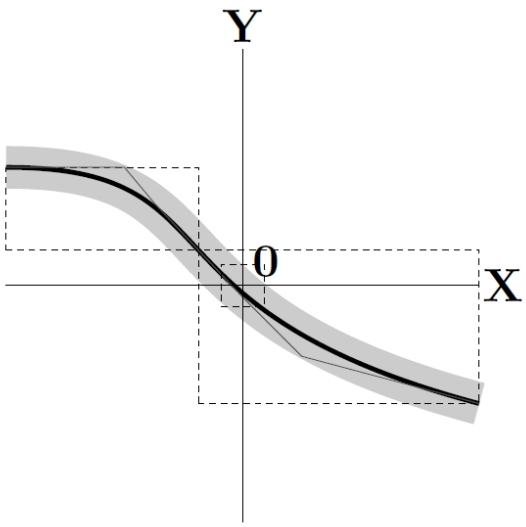
Obrázek 17: Nákres projekce vlákna (křivky) do roviny, která je kolmá ke směrovému vektoru sledovaného paprsku

zanoření společnou pro všechny křivky. Při nižších hodnotách (v intervalu $\langle 1, 5 \rangle$) docházelo u více nelineárních křivek ke špatnému vyhodnocení kolizního testu. Naopak při hloubce zanoření velikosti deset proběhly všechny sledované testy správně. Tento přístup se však neukázal jako výhodný z důvodu větší časové náročnosti, než mělo počítání specifické hloubky zanoření pro každou křivku zvlášť. Hlavním důvodem je, že téměř lineární křivky nepotřebují tak velkou hloubku zanoření potřebnou pro jejich realistickou interpolaci úsečkami.

Nyní následuje rekurzivní dělení křivky. Ilustrace tohoto procesu je na obrázku 11. V rámci rekurzivní procedury se nejdříve určí obalový kvádr křivky (bližší informace v nadcházející podkapitole 5.2.1). Poté se provede test, zdali obdélník tvořený x-ovou a y-ovou souřadnicí obalového kvádru obsahuje čtverec se středem na souřadnicích $(0, 0)$ a s délkou strany rovnou šířce křivky (vlákna). Pokud tento čtverec neobsahuje, tak průsečík paprsku s křivkou naleze najít. Tato situace je znázorněna na obrázku 18. Za předpokladu, že nebylo dosaženo maximální hloubky zanoření, se provede rozdělení křivky (bližší informace v podkapitole 5.2.2) na dvě poloviny. Poté se dekrementuje hloubka zanoření a pro obě poloviny této křivky se zavolá rekurzivní procedura.

Jakmile je dosažena maximální hloubka rekurze je křivka nahrazena úsečkou (vektorem), která je určena prvním a posledním řídícím bodem segmentu křivky. Důležité je nastavit z-ovou souřadnici obou bodů na hodnotu nula, protože po transformaci křivky do dvourozměrného prostoru je tato souřadnice nenulová. Na tento detail jsem poměrně dlouho a složitě přicházel. Algoritmus může bez tohoto kroku fungovat, ale jen v určitých situacích a navíc nepředvídatelně.

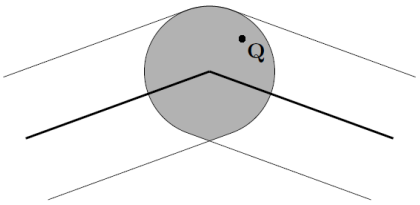
Poté následuje test, jestli výsledný průsečík nepochází z obou segmentů vzniklých po rekurzivním dělením té samé křivky (situace je zachycena na obrázku 20a). Tato situace může ve výsledném obraze způsobit nežádoucí artefakty. Tento test probíhá tak, že jsou v obou krajních bodech segmentu křivky určeny tečné vektory. Pro Bézierovi křivky jsou



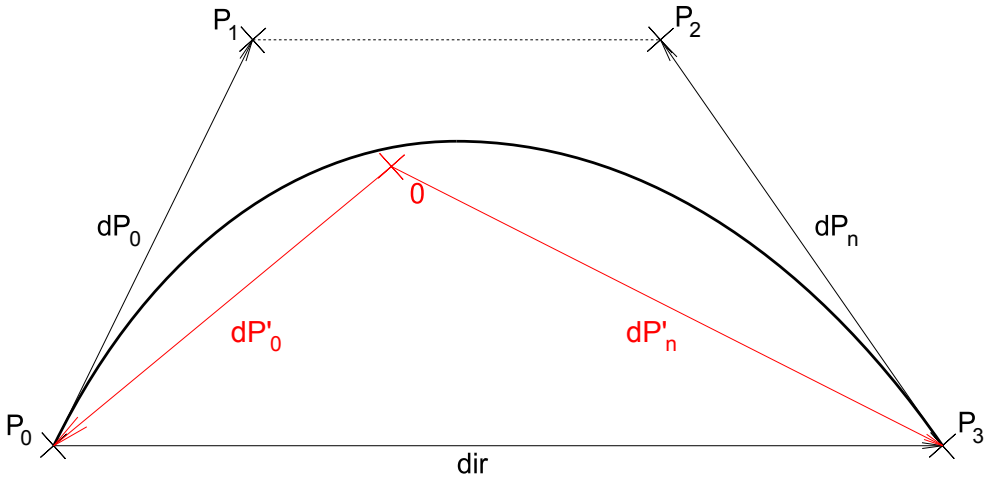
Obrázek 18: Křivka po jednom rekurzivním dělení. Test, zdali obdélník tvořený x-ovou a y-ovou souřadnicí obalového kvádru obsahuje čtverec se středem na souřadnicích $(0, 0)$ a s délkou strany rovnou šířce křivky (vlákna). Na obrázku je možné pozorovat, že další dělení levého segmentu křivky již nemá smysl. Zdroj: [14]

tyto vektory určeny jako rozdíl krajního řídícího bodu a jeho sousedního řídícího bodu. Směr výsledného vektoru je určen podle znaménka jeho skalárního součinu s úsečkou (vektorem), která vznikla interpolací segmentu křivky (dále směrového vektoru křivky) a to tak, aby směr tečného vektoru v prvním řídícím bodě ($\mathrm{d}\mathbf{P_0}$) byl stejný jako u směrového vektoru křivky (**dir**) a směr tečného vektoru v posledním řídícím bodě $\mathrm{d}\mathbf{P_n}$ byl opačný než směrový vektor křivky (na obrázku 20b je nákres situace). Nakonec je testován skalární součin vektorů $\mathrm{d}\mathbf{P_0}$ se záporným vektorem vedoucím z bodu $(0, 0)$ k prvnímu řídícímu bodu $-\mathrm{d}\mathbf{P'_0}$. Pokud je tento skalární součin menší než nula, tak průsečík není nalezen. Obdobně je testován i skalární součin vektorů $\mathrm{d}\mathbf{P_n}$ s vektorem vedoucím z bodu $(0, 0)$ k poslednímu řídícímu bodu $\mathrm{d}\mathbf{P'_n}$. Pokud je výsledek menší nula průsečík opět není nalezen.

Dalším krokem je určení parametru w, který slouží jako parametr pro parametrické vyjádření křivky. Parametr w odpovídá bodu ležícímu na segmentu křivky, který je nejblíže bodu $(0, 0)$ (tedy nejblíže k paprsku, tato situace je zobrazena na nákresu (13) v transformovaném souřadném systému. Hodnota parametru w se vypočítá prostým dosazením do vzorce 3. Při implementaci bylo akorát zapotřebí přidat ořezání výsledného parametru na hodnoty v intervalu $\langle 0, 1\rangle$. Dále bylo potřeba kontrolovat, jestli jmenovatel nenabývá hodnotu nula. Načež je algoritmem de Casteljau (znázorněn na obrázku 14) pomocí parametru w z transformovaných řídících bodů vypočítána poloha bodu (P') na segmentu křivky. Tento bod leží nejblíže k bodu $(0, 0)$ v transformovaném souřadném systému. Následně probíhá kontrola, jestli se bod $(0, 0)$ nachází ve vzdálenosti šířky křivky (vlákna) od vypočteného bodu. Pokud ano znamená to, že existuje průsečík této křivky a sledo-



(a)



(b)

Obrázek 19: Na prvním obrázku (a) (Zdroj: [14]) je zobrazena problémová situace, kdy by mohli vzniknout dva průsečíky místo jednoho kvůli překrývajícím se sousedním segmentům. Na druhém obrázku (b) je nákres prvků potřebných pro testování, zdali tato situace nastala

vaného paprsku. Nyní je potřeba určit parametr v, který odpovídá stejnému bodu jako parametr w, akorát v rámci celé původní křivky. Hodnota parametru v je určena pomocí lineární interpolace parametrů rekurzivní funkce v_0 a v_n, které odpovídají parametrům původní křivky v krajních řídících bodech aktuálně zpracovávaného segmentu křivky. Pomocí parametru v a původních souřadnic celé křivky před transformací do 2D je vypočten bod (P) na ose křivky, který je nejblíže k průsečíku paprsku s křivkou. Situaci si lze představit jako střed průřezu vlákna, jehož rovina je kolmá na tečný vektor křivky v bodě, kde došlo ke kolizi (na obrázku 21 je zobrazena tato situace). Jako v předchozím případě probíhá opět výpočet tohoto bodu algoritmem de Casteljau. Tento vypočítaný bod je potřebný k tomu, abychom dokázali určit vzdálenost od bodu pozorování (počátek zpracovávaného paprsku) k bodu průsečíku v souřadném systému scény. Nakonec jsou do sledovaného paprsku uloženy informace potřebné pro výpočet normály povrchu

křivky v bodě kolize. Popis výpočtu normály křivky (vlákna) v průsečíku bude rozebrán v následující kapitole.

5.2.1 Obalový kvádr křivky

Obalový kvádr segmentu křivky je zapotřebí pro rychlé určení, zdali při výpočtu kolizního testu paprsku a segmentu křivky může existovat průsečík. Pro účely tohoto testu stačí teoreticky jakékoliv konvexní těleso, které zaobaluje kompletně křivku. Avšak pro ideální účinnost tohoto testu je vhodné, aby toto těleso bylo ideálně minimálním konvexním obalem segmentu křivky. Dále je z důvodu porovnání přesahu bodu, který reprezentuje průsečík paprsku s rovinou, na které je promítnuta křivka vhodné, aby toto těleso bylo rovnoběžníkem. V takovémto případě samotné porovnání spočívá v pouhém porovnání patřičných souřadnic. Tento přístup je velice výhodný zejména proto, že tento test nezabere mnoho času a zároveň značně urychlí celkový čas na určení bodu kolize tím, že v čas eliminuje segmenty křivky, které nemohou mít průsečík se zpracovávaným paprskem. Způsobů jak toto obalové těleso sestrojit existuje několik. Využil jsem toho, že křivky v mé aplikaci jsou reprezentovány Bézierovými křivkami třetího řádu. Tyto křivky mají příhodnou vlastnost a to že všechny body křivky leží uvnitř konvexního obalu tvořeného jejími řídícími body. Vytvořil jsem jednoduchou metodu, která vytváří obalové těleso ve tvaru kvádru, který je reprezentován dvěma body. První bod (dále minimální bod) je z pohledu souřadného systému ten, který má všechny souřadnice nejmenší a druhý bod (dále maximální bod) je zase ten, který má souřadnice největší. Z počátku je minimální bod inicializován na velkou hodnotu (například maximální hodnotu typu float na dané architektuře) a maximální bod je inicializován na velice nízkou hodnotu (opět například minimální hodnotu typu float na dané architektuře). Poté se prochází všechny řídící body křivky, pro kterou se má obalové těleso vytvořit. Dále je pro všechny souřadnice řídících bodů provedeno porovnání, jestli je tato hodnota menší, než hodnota dané souřadnice minimálního bodu. Pokud ano, tak je hodnota souřadnice minimálního bodu přepsána hodnotou souřadnice řídícího bodu. Naopak, pokud je hodnota souřadnice větší než hodnota dané souřadnice maximálního bodu, tak je hodnota souřadnice maximálního bodu nastavena na hodnotou souřadnice onoho řídícího bodu. Tímto způsobem lze vytvořit pro jakoukoliv Bézierovu křivku třetího řádu obalové těleso ve tvaru kvádru. Chtěl bych podotknout, že takto vytvořené obalové těleso nemusí být a často také není minimální. Na druhou stranu je tato metoda velice rychlá a pro účely zvoleného algoritmu pro výpočet kolizního testu paprsku a křivky dostačující. Je pravděpodobné, že by použití jiného algoritmu, který dokáže vypočítat ve stejném nebo mírně horším čase optimální obalové těleso a zároveň porovnat, jestli toto těleso obsahuje bod reprezentující průsečík paprsku s rovinou, na které je promítnuta zpracovávaná křivka, kladně ovlivnilo čas na zpracování jednoho kolizního testu.

5.2.2 Dělení křivky

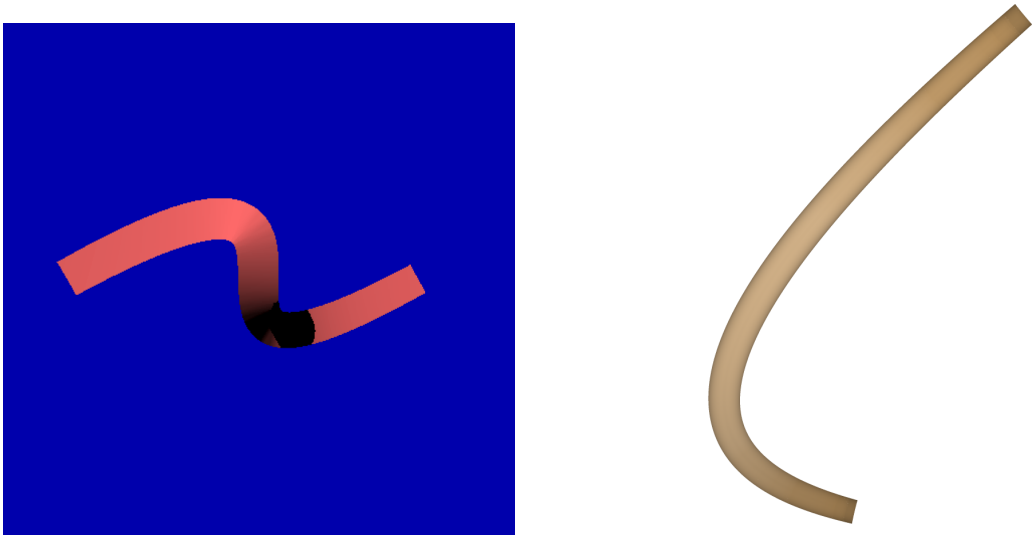
Vždy před rekurzivním zanořením je zapotřebí během algoritmu operace dělení křivky. Aby se správně vyhodnocoval test (popsán výše v podkapitole 5.2) na odstranění pro-

blému s nalezením průsečíků v obou přilehlých koncích rozdělené křivky (tento jev způsobuje nežádoucí alias na zobrazovaném vlákně), je zapotřebí splnit jednu podmínku. Tato podmínka spočívá v tom, aby na žádné části křivky po rozdělení nebyl inflexní bod. Tedy bod, ve kterém se křivka mění z konkávní na konvexní nebo obráceně. Ovšem pro Bézierovy křivky nižších řádů je tento požadavek téměř zanedbatelný. Empiricky jsem vyhodnotil, že pokud se křivka při každém dělení dělí na poloviny, tak se ve výsledném obraze neprojevují zřetelné aliasy způsobené touto problematikou. Nyní k samotnému dělení křivky. Dělení křivky na dvě části, kde je bod rozdělení definován parametrem pro parametrické vyjádření křivky, je pro Bézierovi křivky velice vhodný algoritmus de Casteljau a to kvůli následujícímu důvodu. Algoritmus de Casteljau při hledání bodu křivky, který odpovídá danému parametru, rekurzivně vyhledává (určuje) pomocí lineární interpolace pomocné body na úsečkách tvořených sousedními řídícími body a poté vyhledává další pomocné body na úsečkách tvořenými již nalezenými pomocnými body. Takto vzniká pět nových bodů. Pokud hledaný bod odpovídá bodu, ve kterém se má křivka rozdělit, tak první původní řídící bod, první a druhý pomocný bod a hledaný bod (na obrázku 12 jsou to body P_0, $P_{1,1}$, $P_{2,2}$ a $P_{3,3}$) tvoří řídící body první části dělené křivky. Druhou část křivky tvoří hledaný bod, pátý a čtvrtý pomocný bod a čtvrtý původní řídící bod (na obrázku 12 jsou to body $P_{3,3}$, $P_{3,2}$, $P_{3,1}$ a P_3). Dělení křivky pomocí upraveného algoritmu de Casteljau je znázorněno na obrázku 12.

5.3 Výpočet normál křivek

Výpočet normály povrchu vlákna v místě, kde bylo zasaženo vyšetřovaným paprskem je nepostradatelnou informací pro výpočet osvětlení vlákna v daném bodě. Určení normály nebylo příliš přímočaré, hlavně kvůli zvolenému algoritmu pro výpočet kolizních testů paprsků a křivek. Dále také z důvodu transformace křivky do dvourozměrného prostoru a také kvůli zaokrouhlovacím chybám způsobených velice malou geometrií vláken. Autor metody pro výpočet průsečíků zvolil výpočet normály křivky (vlákna) tak, jako by to byl pásek (povrch je rovinný a ne oblý). Tím ušetřil výpočetní čas, ale bohužel detailní kvalita zobrazené křivky působí velice nerealisticky. Na obrázku 20 můžete vidět, jak vypadá stínovaná křivka (vlákno), pokud jsou normály vlákna počítány jako by to byl pásek. Jelikož mou snahou je, aby výsledný vzhled obrázku po vytvoření byl co nejrealističtější, musel jsem vymyslet vlastní výpočet normál. Nejprve jsem provedl úvahu, že znám bod (P) na ose křivky, který je nejblíže k bodu, ve kterém paprsek protíná tuto křivku. Tento bod je vypočten pomocí parametru křivky v z celé původní křivky před transformací a rekurzivním dělením, způsob výpočtu tohoto bodu je popsán v kapitole 5.2. Pro zpřehlednění budu netransformované souřadnice nazývat světovými souřadnicemi. Dále znám souřadnice bodu, který odpovídá tomuto bodu po transformaci křivky. Tento bod (P') je vypočten pomocí parametru křivky w ze segmentu křivky, který je již po transformaci. Způsob výpočtu tohoto bodu je popsán opět v kapitole 5.2. Dále vím, že ve transformovaných souřadnicích protíná paprsek v bodě $(0, 0)$ rovinu, na které leží bod (P'). Tato rovina má normálový vektor odpovídající směrovému vektoru paprsku (je na něj kolmá). Z těchto dvou bodů zjistím jejich vzdálenost d , která je odvěsnou v trojúhelníku, ve kterém je přepona délky rovna poloměru vlákna r a druhá odvěsna určuje vzdálenost δt o kterou je

potřeba zkrátit vzdálenost mezi bodem P a počátečním bodem sledovaného paprsku tak, aby odpovídala vzdálenosti k reálnému průsečíku paprsku s povrchem vlákna. Nákres této situace je možné vidět na obrázku 21. Jakmile zjistím souřadnice reálného průsečíku paprsku s povrchem vlákna ve světových souřadnicích, je výpočet normály povrchu vlákna v daném bodě roven rozdílu tohoto bodu a bodu P.

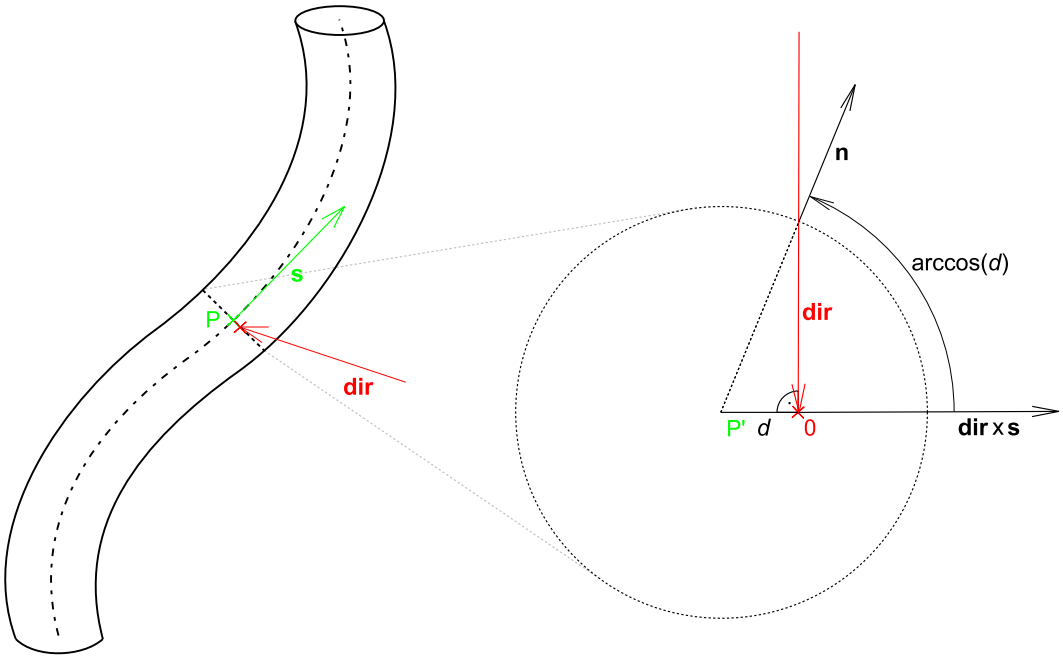


(a) (b)

Obrázek 20: Na prvním obrázku (a) (Zdroj: [14]) je zobrazeno jedno stínované vlákno, kde je výpočet normál počítán, jakoby to byl pásek. Na druhém obrázku je zobrazeno stínované vlákno mou demonstrační aplikací, kde jsou normály počítány postupem uvedeným v kapitole 5.3

Tento postup kupodivu nefungoval a výsledky byly velice neuspokojivé. Důvodem byla nejspíš velice tenká geometrie vláken, při které je výpočet přesného bodu průniku velice náchylný k výpočetním chybám. Proto jsem musel vymyslet jiné řešení a po několika úvahách jsem přišel na následující postup. Nejdříve jsem určil tečný vektor křivky v bodě P ve světových souřadnicích a to tak, že jsem vypočítal bod, který je velice blízko bodu P na ose vlákna. Tento bod jsem získal pomocí parametru v, ke kterému jsem přičetl velice malé číslo a pokud by byl součet větší než jedna, tak je potřeba ho odečíst a rozdíl pro určení vektoru obrátit. Tento vektor je také znormalizován. Poté jsem provedl úvahu, že pokud udělám vektorový součin vypočteného tečného vektoru se záporným směrovým vektorem sledovaného paprsku, tak dostanu vektor, který bude ležet ve stejné rovině jako hledaný normálový vektor. Takže stačí tento vektor patřičně rotovat kolem tečného vektoru křivky o úhel daný arcus cosinem vzdálenosti bodu $(0, 0)$ a bodu P' v transformovaném souřadném systému. Nákres výpočtu je zachycen na obrázku 21. Tento výpočet normály již není tak náchylný k výpočetním chybám, jelikož normálový vektor je vždy ve správném takzvaném normálovém disku a jeho rotace nemusí být

vypočtena s maximální přesností.



Obrázek 21: Ilustrace prvků potřebných pro výpočet normály povrchu vlákna v bodě průsečíku s paprskem. Na levé straně je ve světových souřadnicích zobrazena směrnice osy vlákna **s** (tečna) v bodě P, který je nejblíže k průsečíku paprsku s vláknem. Na pravé straně je detail průřezu vlákna v bodě dopadu paprsku podle roviny, jejíž normála odpovídá směrnici osy vlákna. Bod $(0, 0)$ a P' je zde v transformovaných souřadnicích použit pro výpočet jejich vzdálenosti d. Dále je zde zakreslen vektor vzniklý vektorovým součinem směrového vektoru paprsku **dir** s vektorem **s**. Výsledný vektor je poté rotován okolo osy dané vektorem **s** a to o úhel vypočtený jako $\arccos(d)$. Výsledkem rotace je hledaný normálový vektor **n** povrchu vlákna v bodě dopadu paprsku

5.4 Osvětlovací model

Pro výpočet odraženého světla z povrchu objektů jsem zvolil empirický Phongův osvětlovací model (popsán v publikaci [16]). I když tento model není fyzikálně zcela přesný, přesto lze jeho prostřednictvím dosáhnout velmi realisticky vypadajících výsledků a to při poměrně krátkém čase na výpočet. Výpočet Phongova osvětlovacího modelu pro povrch vlákna je prakticky stejný jako u výpočtu osvětlovacího modelu trojúhelníku. Je tedy zapotřebí znát pouze průsečík zpracovávaného paprsku s geometrickým primitivem a jeho texturu (ambientní, difúzní a spekulární složku barvy), normálu povrchu v bodě průsečíku a polohu všech zdrojů světla. V mém případě jsem zvolil všesměrové zdroje světla. Jedním rozdílem je způsob získávání potřebných informací. Přičemž největší roz-

díl je ve výpočtu normál, který u trojúhelníků probíhá lineární interpolací normál v jejich vrcholech. Postup výpočtu normál vláken je popsán detailně v předchozí podkapitole 5.3.

Dále je zde drobný rozdíl ve způsobu získání patřičné barvy z textury, který u trojúhelníků probíhá opět lineární interpolací texturovacích dat v jejich vrcholech. Abych vytvořil texturu, která není konstantní po celé délce vláken, jsem využil toho, že jsou vlákna specifikovány parametrickými křivkami. Vytvořil jsem strukturu, která obsahuje seznam textur a seznam hodnot parametrů pro parametrické vyjádření křivek. Každá křivka má jedny texturovací souřadnice, což je velice výhodné pro jejich modelování a odkaz na strukturu textur. Při hledání patřičných texturovacích informací pro povrch vlákna je zapotřebí znát parametr pro parametrické vyjádření bodu, který leží na křivce (pomyslné ose vlákna) nejblíže průsečíku. Tento bod známe již z hledání onoho průsečíku (bod P, jeho výpočet je popsán v kapitolách 5.2 a 5.3). Nejprve jsou nalezeny barvy z jednotlivých textur, které odpovídají daným texturovacím souřadnicím. Poté vezmu seznam parametrů , ve kterých jednotlivé textury určují barvu a stochasticky změním jejich hodnotu, tak aby všechna vlákna neměnily barvu ve stejných délkách. Následně zjistím, mezi kterými parametry leží hledaný parametr. Na závěr provedu lineární interpolaci texturovacích informací odpovídající sousedním parametrům. Princip je podobný jako u volumetrických textur.

Vytváření stínů probíhá obdobně jako u trojúhelníků, jelikož jde opět jen o hledání průsečíků stínových paprsků, které mají počátek v bodě, ve kterém se ověřuje zastínění, přičemž směrový vektor je určen rozdílem bodu, ve kterém se nachází zdroj světla a počátek paprsku. Bohužel zde vyplynul jeden problém, na který jsem již narazil u výpočtu normál. Transformace geometrie a výpočetní chyby způsobily to, že vypočtený průsečík jen pomocí paprsku a parametru t udávajícího vzdálenost průsečíku od počátku paprsku není zcela přesný. K docílení uspokojivějších výsledků jsem místo takto vypočteného průsečíku použil bod P (popsaný v předchozí kapitole 5.3) s přičtenou šířkou vlákna ve směru zdroje světla jako počátek stínového paprsku. Toto řešení není fyzikálně zcela přesné, ale trpí mnohem menší výpočetní chybou a poskytuje výsledky bez nevhodných stínových aliasů.

5.5 Akcelerace pomocí hierarchické struktury

Jelikož se v rámci této práce zaměřuji hlavně na vizuální kvalitu, nedal jsem z počátku implementaci hierarchické akcelerační struktury příliš vysokou prioritu. Avšak testování zobrazovacího algoritmu na komplexnějších scénách bylo poněkud zdlouhavé. Přemýšlel jsem, jak ušetřit čas čekání na výstupy a zároveň neztratit příliš mnoho času implementací a poměrně komplikovaným laděním velice sofistikované akcelerační struktury, která je popsána v kapitole 4.4.1. Došel jsem k nápadu, že bych mohl zkusit použít velice populární a často používanou akcelerační strukturu využívanou pro geometrická primitiva ve tvaru trojúhelníků zvanou hierarchie obalových těles, zkráceně BVH (bounding volume hierarchy). Konkrétně pro osově zarovnané obalové tělesa, zkráceně `AABB` (axis aligned bounding box). K vytvoření `AABB` jsem využil již implementované metody k vytvoření osově zarovnaného obalového tělesa pro zvolenou reprezentaci vláken, kterou jsem dříve vytvořil za účelem kolizního testování (popsanou v podkapitole 5.2.1). Poté jsem vytvořil

reprezentaci hierarchické stromové struktury, kde má každý uzel vlastní `AABB` vytvořený postupným rozšiřováním o `AABB` křivek, které daný uzel zaobaluje, odkaz na jeho dva potomky. Pokud je tento uzel listem, tak navíc obsahuje seznam odkazů na patřičné křivky. Dále jsem vytvořil rekurzivní metodu k jeho naplnění. Na počátku je seznam křivek. Nejdříve je vytvořen kořen stromu, jeho `AABB` a je zvolena osa podle které se budou třídit křivky. Následovně je seznam křivek částečně seřazen podle x-ové souřadnice a to v lineárním čase pomocí standardní C++ funkce `std::nth_element`, která pomocí modifikovaného quick sortu určí prvek, který leží na n-té pozici tak, aby prvky, které mají menší nebo stejnou hodnotu ležely na jedné straně a prvky s vyšší nebo stejnou hodnotou na druhé straně. Takže při určení n jako prostředního indexu seznamu, jsou křivky vhodně setříděné k rozdělení na dva seznamy. Tyto seznamy se poté předají rekurzivnímu volání této metody, přičemž jsou výstupy těchto metod navázány na potomky kořene. Rekurze končí, pokud je seznam prvků menší než vhodně zvolená konstanta. Zvolil jsem, že každý list bude obsahovat odkaz na tři křivky.

Jakmile jsem na implementoval reprezentaci a metody k vytvoření BVH pro křivky, chyběla už jenom metoda pro vyhledání křivek, pro které potenciálně existuje průsečík s daným paprskem. Pokud tento průsečík existuje, tak ho lze pomocí kolizního testu paprsku a křivky nalézt. Strom je prohledáván pro daný paprsek rekurzivně. Nejdříve se provede kolizní test paprsku s osově zarovnaným obalovým tělesem kořene stromu a pokud existuje průsečík, tak je metoda volána rekurzivně pro oba synovské uzly. Pokud je testovaný uzel listem, provede se kolizní testování všech křivek, na které má list stromu odkaz. Tímto krokem je zjištěno, jestli existuje nějaký průsečík křivky se zpracovávaným paprskem pro danou větev BVH. Při vynořování z rekurze se vynořuje nejbližší průsečík k bodu počátku sledovaného paprsku, pouze pokud existují zjištěné průsečíky pro více větví stromu. Výpočet kolizního testu paprsku a AABB je relativně přímočará záležitost a lze použít například algoritmu popsaném v publikaci [27]. Zrychlení po aplikaci akcelerační struktury bylo poměrně uspokojivé, hlavně pro scény obsahující větší počet vláken. Podrobnější výsledky budou popsány v kapitole 6.

5.6 Začlenění do ray traceru

Jeden z hlavních důvodů, proč jsem zvolil tento konkrétní algoritmus pro zobrazování křivek metodou sledování paprsků byl, že je jeho přístup k hledání průsečíků a celkový přístup k výpočtu je tvořen tak, aby se příliš nelišil od výpočtů běžných pro raytracing trojúhelníků. Na rozdíl od přístupů jako je například clone tracing, na kterém je založena metoda Cone Tracing for Furry Object Rendering (popsána v kapitole 3.1). Proto bylo relativně snadné začlenit algoritmus do tradičního frameworku raytraceru a rozšířit zobrazování například o nad-vzorkování (multisampling) nebo o výpočet hloubky ostrosti (kamerového rozostření) a nebylo problémem aplikaci rozšířit o ray tracing trojúhelníků. Dokonce by bylo možné mnou zvolenou akcelerační strukturu (popsanou v předchozí podkapitole) sjednotit pro obě geometrické primitiva a přistupovat k nim polymorfně. Nepřineslo by to sice příliš velké časové úspory, ale zato je na tomto příkladu dobře vidět, jak je algoritmus začlenitelný. Také by bylo možné dokonce díky mnou zvolené akcelerační struktuře popsané v předchozí podkapitole ji sjednotit pro obě primitiva a přistupovat

k nim polymorfně, nepřineslo by to příliš velké časové úspory, ale je na tomto příkladu hezky vidět, jak je algoritmus začlenitelný.

5.7 Testovací modely

V této kapitole stručně popíši reprezentaci a tvorbu modelů scén pro účely testování algoritmu. Jako reprezentaci modelu scény jsem vytvořil strukturu, která obsahuje seznamy všech trojúhelníků a Béziérových křivek scény. Dále obsahuje patřičné textury pro trojúhelníky a speciální textury pro Béziérovi křivky (vlákna), které jsou vytvořeny za účelem simulace podobného efektu jako mají volumetrické textury (způsob použití je popsán v podkapitole 5.4). Dále ještě obsahuje sadu speciálních textur pro cílenou modifikaci konkrétních řídících bodů vláken k realističtější simulaci některých scén, tuto funkcionalitu detailněji popíši níže.

Zvolil jsem procedurální tvorbu modelů, ale nebyl by příliš velký problém rozšířit aplikaci také o používání modelů vytvořených pomocí programů třetích stran. Bylo by jen zapotřebí převést reprezentaci křivek importovaných modelů do zvolené interní reprezentace. Pokud by byly křivky v importovaném modelu reprezentovány jinými než Bézierovými křivkami třetího řádu, bylo by nutné implementovat funkci na převod křivek do této podoby. Pro většinu používaných křivek tento postup existuje a pokud ne tak lze zvolit nějakou vhodnou aproximaci.

V tomto odstavci zhruba popíši, jak jsem vytvářel konkrétní testovací modely. Model trávníku (obrázky 25, 22) jsem vytvořil tak, že jsem na ploše čtverce vygeneroval parametrem daný počet řad a sloupců rovných vláken. Přitom jsem počátek vlákna stochasticky posouval tak, aby nevznikla uniformní síť. Poté jsem opět stochasticky měnil polohu ostatních řídících bodů (teď už ve směru všech os) s předem danou maximální velikostí náhodného posunu. Každé vlákno má jednu texturovací souřadnici, která je dána dvourozměrnou souřadnicí čtverce podstavy v bodě, kde se vytvořil počáteční řídící bod vlákna. Po vytvoření všech vláken je ještě provedena modifikace řídících bodů všech vláken podle struktury, kterou jsem nazval směrnicovou mapou. Ta je tvořena čtveřicí textur (pro každý řídící bod jedna) složených z jednoduchých barev. V rámci mapy jsem definoval pravidlo pro každou použitou barvu. Například červená barva znamená, že se řídící body, které mají texturu odpovídající červené barvě posunou v předem definovaném směru a například zelená barva bude znamenat, že se mají posunout v jiném směru. Poté jsem ještě doplnil pravidlo, aby se velikost míry posunutí v daném směru zvětšovala pro řídící body, které jsou dál od prvního řídícího bodu. Tímto způsobem lze docela elegantně simulovat například ohyb trávy vlivem větru ve skupinách stébel.

Model jednoho pramene vlasů, který lze pozorovat na obrázku 24, jsem vytvořil velice podobným způsobem. Opět jsem z počátku vytvořil předem specifikovaný počet rovných vláken na čtvercové podstavě a poté jsem všechny tyto vlákna prošel a modifikoval jejich poslední (nejvzdálenější od počátku) řídící bod tak, abych ho posunul ve směru vektoru určeným rozdílem prvního řídícího bodu vlákna uprostřed podstavy a prvního řídícího bodu vynásobeného o vhodně zvolenou konstantu. Poté akorát modifikuji délku vlákna tak, aby byla v konstantní vzdálenosti od prvního řídícího bodu středního vlákna a vytvořila tímto způsobem ono rozložení viditelné na obrázku.

U modelu copu je každý pramen tvořen třemi Bézierovými křivkami a to tak, že je každý pramen vytvořen zvlášť s tím, že počáteční podstava pramene vlasů je posunuta doleva či doprava a u druhého pramene naopak. Navíc je zde posunuta hloubková složka k vytvoření efektu zapletení pramenů do sebe. Tuto scénu je možné vidět na obrázku 27.

Model koule pokryté vlákny je vytvořen pomocí rotace quaternionů reprezentujících směrové vektory jednotlivých vláken. Mapování textury souřadnic (u, v) je provedeno procedurálně na základě vztahů:

$$u = 0,5 + \frac{\arctan 2(d_z/d_x)}{2\pi}\,,$$
$$v = 0,5 - \frac{\arcsin 2(d_y)}{\pi}\,. \tag{28}$$

Kde **d** je normovaný vektor směřující z prvního řídícího bodu vlákna do středu koule. Dále jsou jednotlivé řídící body stochasticky modifikovány a mohou být také ovlivňovány směrnicovými mapami. Vytvořil jsem ještě několik testovacích modelů, které vycházejí z výše popsaných modelů a jsou jejich drobnou modifikací. V následující kapitole jsou shrnuty výstupy vytvořené demonstrační aplikací a jejich zhodnocení, zejména vizuální kvality.

6 Vyhodnocení výsledků

Moji demonstrační aplikaci jsem testoval především na počítači s dvou jádrovým procesorem Intel Core i5-4200M s taktem 2,50GHz pod 64-bitovým operačním systémem Windows 7, přičemž jsem aplikaci paralelizoval na CPU pomocí vláken. Výsledky časových testů jsem zaznamenal do tabulky 1. Pro určení jednotlivých časů jsem použil funkci OpenMP `omp_get_wtime`. Při testování jsem se zaměřil hlavně na srovnání časů pro zobrazení scény s modelem trávníku s různými počty vláken. Sledoval jsem také časy potřebné na vytvoření použité hierarchické akcelerační struktury (BVH) popsané v podkapitole 5.5. Dále jsem sledoval vztah mezi dobou výpočtu nejdříve s použitím a poté bez použití BVH. Z těchto testovacích časů lze pozorovat, že použití BVH přináší velkou úsporu času a doba potřebná k vytvoření BVH je v poměru k celkovému času na vykreslení scény téměř zanedbatelná. Také jsem zaznamenal časy pro vytvoření obrazu v různých rozlišeních a to při aplikaci různých počtů vzorků pro supersampling (nadvzorkování).

Model trávníku		512×512 bez SS	512×512 3×3 SS	1024×1024 bez SS	1024×1024 3×3 SS
256 vláken	tvorba BVH	0,6 ms	0,7 ms	0,5 ms	0,6 ms
	vykreslení s BVH	18,7 s	2,8 min	1,5 min	13,1 min
	vykreslení bez BVH	48,6 s	7,4 min	3,5 min	31,1 min
441 vláken	tvorba BVH	0,5 ms	1,4 ms	0,7 ms	1,0 ms
	vykreslení s BVH	29,8 s	4,5 min	2,4 min	18,9 min
	vykreslení bez BVH	1,4 min	12,9 min	5,9 min	52,4 min
961 vláken	tvorba BVH	4,6 ms	14,4 ms	2,6 ms	10,4 ms
	vykreslení s BVH	57,9 s	8,9 min	3,8 min	34,4 min
	vykreslení bez BVH	30,1 min	28,7 min	13,1 min	1,9 h
10 201 vl.	tvorba BVH	384,7 ms	251,5 ms	393,4 ms	620,7 ms
	vykreslení s BVH	5,5 min	49,0 min	23,3 min	3,7 h
	vykreslení bez BVH	35,6 min	5,4 h	2,4 h	21,0 h

Tabulka 1: Tabulka zachycuje časy potřebné na vytvoření hierarchické akcelerační struktury (BVH) a časy potřebné na zobrazení scény s různým počtem vláken s použitím a poté bez použití BVH. Časy jsou sledovány pro různé rozlišení výstupního obrazu (v pixelech) a to při aplikaci různých počtů vzorků pro supersampling (nadvzorkování)

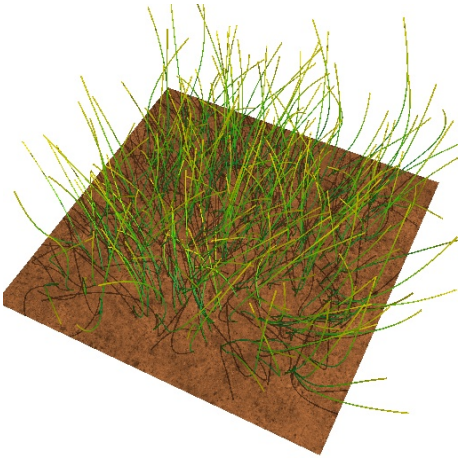
Abych navázal na předpoklad týkající se významu zvolené reprezentace vláken oproti jejich triangulaci, který jsem popsal na konci podkapitoly 5.1. Z tohoto důvodu jsem sledoval průměrný čas na výpočet jednotlivých kolizních testů a naměřené výsledky jsem zachytil v tabulce 2. Z těchto výsledků lze pozorovat, že kolizní test paprsku a křivky je v průměru přibližně třináctkrát pomalejší než kolizní test paprsku s trojúhelníkem. Z toho plyne, že by se jedna křivka (vlákno) musela v průměru triangulovat jen pomocí třinácti trojúhelníků, přičemž výsledek by nebyl příliš vizuálně kvalitní. Celkové časy na zobra-

zení scén jsou přesto poměrně vysoké a to hlavně proto, že renderování probíhá pouze na CPU. Tedy neprobíhá zde žádná akcelerace na straně grafické karty. Tento fakt bylo možné předpokládat, jelikož jsem se v souladu se zadáním této práce měl zaměřit primárně na vizuální kvalitu.

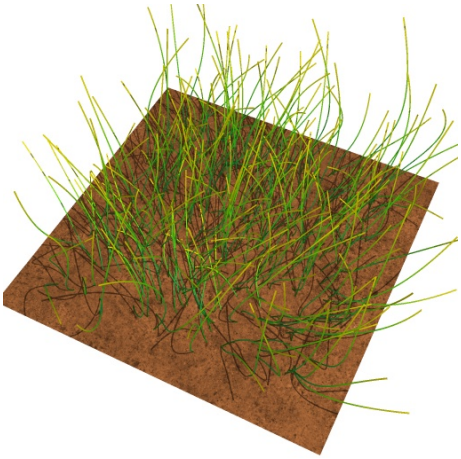
Průměrný čas na výpočet kolizního testu		
paprsek - křivka	paprsek - trojůhelník	poměr
0,575 μs	0,045 μs	12,8

Tabulka 2: Tabulka obsahuje výsledky srovnání testování průměrného času potřebného na vyhodnocení jednoho kolizního testu paprsku s trojúhelníkem nebo křivkou a poměr těchto hodnot. Z měření vyplývá, že kolizní test paprsku s trojúhelníkem je v průměru $12,8\times$ rychlejší než kolizní test paprsku s křivkou. Tyto průměrné hodnoty byl vypočteny z 10000 testů

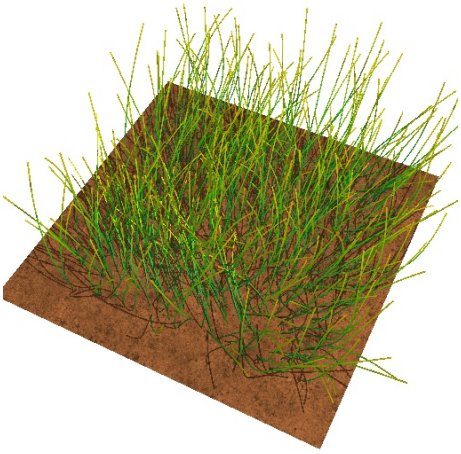
Na následujících obrázcích (22, 23, 24, 25, 26, 27) je možné pozorovat výsledky testovacích scén s různými parametry, které jsou zobrazeny pomocí demonstrační aplikace. Výstupy byly renderovány s Phongovým osvětlovacím modelem (popsaném v publikaci [16]) společně s vytvářením stínů. K docílení realističtějšího vzhledu bylo zapotřebí použít supersampling. Při porovnání s výsledky vytvořenými pomocí algoritmu, na základě kterého demonstrační aplikace vznikla (možné pozorovat na obrázcích 9), bych hodnotil mé výstupy jako uspokojivé. Velice kladně hodnotím vzhled jednotlivých vláken a to především díky časově náročnému výpočtu normál jejich povrchů, který jsem navrhl v kapitole 5.3. V rámci aktuálních přístupů (popsáných v kapitole 3) k zobrazování vlasů a srsti pomocí metody sledování paprsků, jsou vlákna často zobrazovány jako částečně průhledné objekty. Díky tomuto efektu dosahují na vhodných scénách vizuálně líbivějších výsledků. Tento přístup by bylo možné využít jako možné vylepšení stávající demonstrační aplikace.



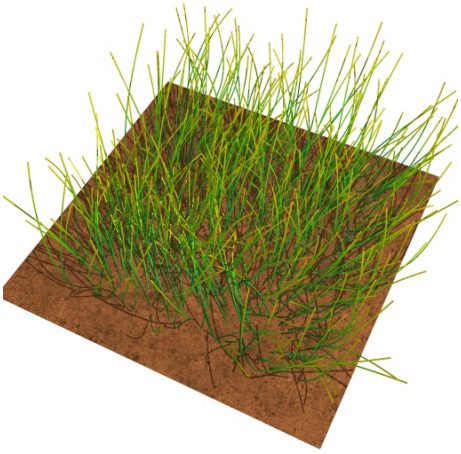
(a) 256 vláken, bez SS



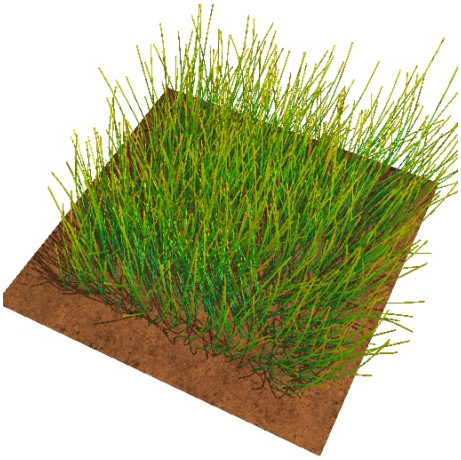
(b) 256 vláken, 3×3 SS



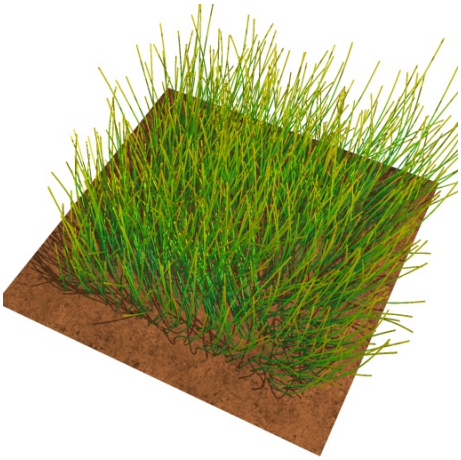
(c) 441 vláken, bez SS



(d) 441 vláken, 3×3 SS



(e) 961 vláken, bez SS



(f) 961 vláken, 3×3 SS

Obrázek 22: Model trávníku zobrazen s různými počty vláken a srovnání aplikace supersamplingu na výsledné zobrazení. Na tomto testovacím modelu lze patrně vidět aplikaci takzvaných směrnicových map popsaných v podkapitole 5.7 v kombinaci s parametrickými volumetrickými texturami. Na modelu půdy je možné pozorovat kvalitu vrhaných stínů. Časy potřebné na vykreslení těchto obrázků jsou uvedeny v tabulce 1

Obrázek 23: Model koule tvořené vlákny s texturou srsti leoparda. Scéna je zobrazena v rozlišení 1024×1024 pixelů při použití 4×4 supersamplingu a obsahuje 97 470 vláken. Vytvoření tohoto obrázku trvalo 8 hodin a 40 minut

Obrázek 24: Model svazku vlasů, který je tvořen 961 vlákny. Scéna je zobrazena v rozlišení 1024×1024 pixelů při použití 3×3 vzorků na supersampling. Vytvoření tohoto obrázku trvalo 30 minut a 12 sekund

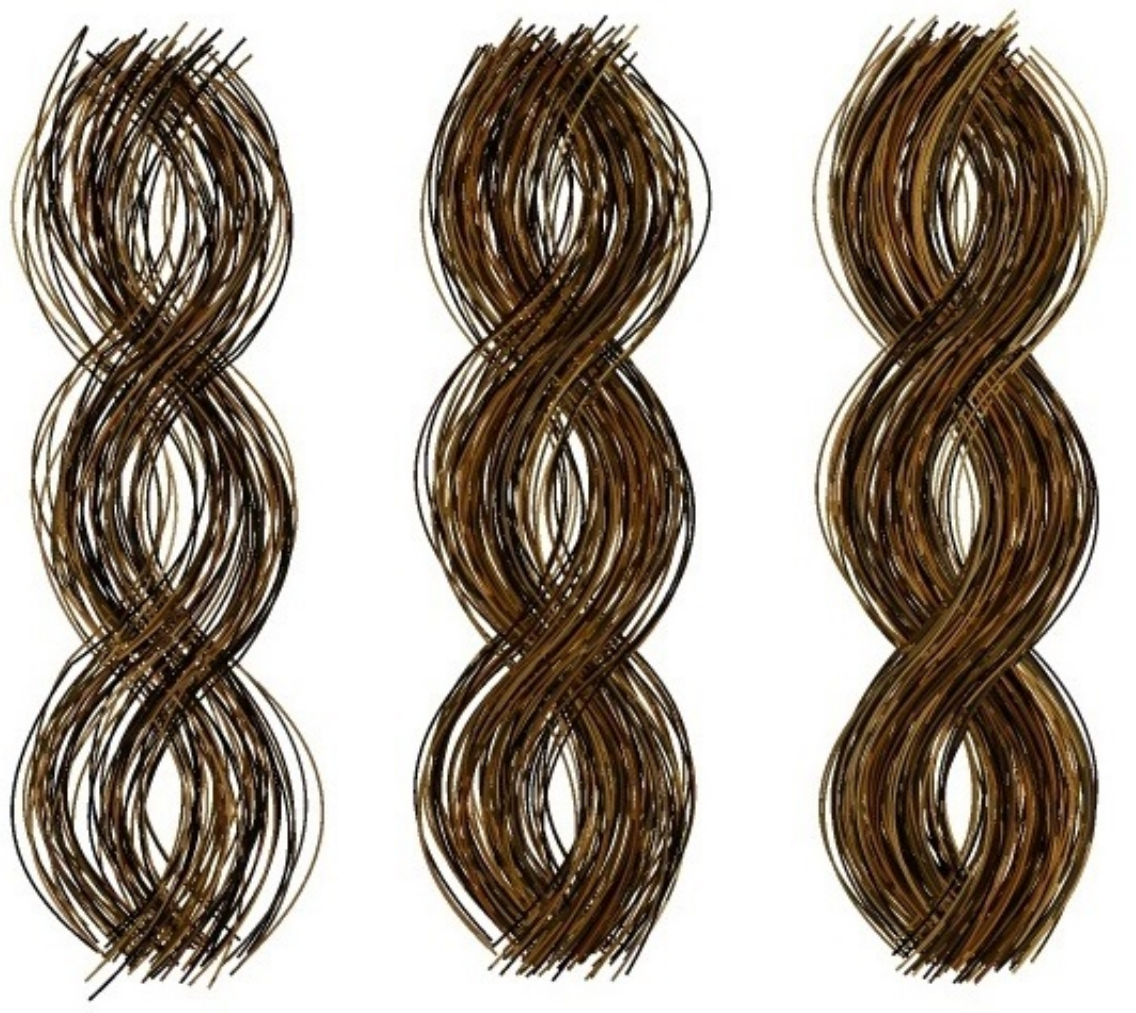
Obrázek 25: Scéna s modelem trávníku v terénu. Model se skládá z 9604 vláken a 19602 trojúhelníků a je renderována v rozlišení 1024×1024 pixelů s 3×3 supersamplingem. Jde o model trávníku, kde je povrch země modifikován gradientní mapou. Vytvoření tohoto obrázku trvalo 2 hodiny a 12 minut

(a) 50 vláken, 1 minuta 8 sekund

(b) 200 vláken, 5 minut 27 sekund

(c) 450 vláken, 12 minut 26 sekund

Obrázek 26: Modely propletených pramenů vlasů s různými počty vláken zobrazených v rozlišení 512×512 pixelů s 3×3 supersamplingem. Časy potřebné na vykreslení jsou uvedeny u jednotlivých obrázků



(a) 294 vláken, 1 minuta 57 sekund

(b) 486 vláken, 2 minuty 59 sekund

(c) 726 vláken, 4 minuty 26 sekund

Obrázek 27: Modely propletených vlasů s různými počty vláken zobrazených v rozlišení 512×512 pixelů s 3×3 supersamplingem. Časy potřebné na vykreslení jsou uvedeny u jednotlivých obrázků

7 Závěr

Cílem této diplomové práce bylo prostudovat a popsat vybrané aktuální algoritmy pro zobrazování vlasů a srsti metodou sledování paprsků. Následně některý z těchto algoritmů detailně rozebrat, implementovat a vyhodnotit dosažené výsledky. V kapitole Přehled používaných metod jsem chronologicky zachytil vývoj a principy metod, které nějakým způsobem ovlivnily aktuální moderní techniky ve zkoumané oblasti. V následující části jsem detailněji popsal tři aktuální techniky vhodné pro raytracing vlasů a srsti. Z těchto technik jsem pro další zkoumání zvolil metodu Distance aware ray tracing for curves a následně ji detailně rozebral v kapitole Algoritmus pro zobrazování vlasů a srsti. Tuto metodu jsem vybral především kvůli její vlastnosti používání geometrie křivek jako reprezentaci vláken, se kterými lze pracovat podobně jako s trojúhelníky. Zejména díky této příhodné vlastnosti je možné metodu použít jako rozšíření tradičního raytraceru, což umožňuje přístup k využití celé řady algoritmů pro různé grafické efekty.

Dále jsem vytvořil demonstrační aplikaci s implementací algoritmu pro zobrazování vlasů a srsti metodou sledování paprsků založenou na zvoleném algoritmu. Za účelem prezentace kvality implementovaného algoritmu jsem vytvořil sadu procedurálně generovaných stochastických testovacích modelů s volumetrickými texturami. V kapitole Vyhodnocení výsledků jsem předvedl zobrazení těchto modelů demonstrační aplikací a zhodnotil zejména jejich vizuální kvalitu, také jsem zde shrnul časy potřebné pro zobrazení scén s různými parametry.

Dosáhl jsem vizuální kvality výstupů srovnatelné s autory zvoleného algoritmu. Při vizuálním srovnání s ostatními aktuálními přístupy v této oblasti bych výsledky mé demonstrační aplikace hodnotil jako uspokojivé. Potenciální rozvoj demonstrační aplikace na plnohodnotný raytracer by byl relativně snadný vzhledem k tomu, že implementovaná metoda je plně kompatibilní s běžnými rozšířeními o grafické efekty, jako jsou například rozostření a rozmazání pohybem. Časy demonstrační aplikace na vytvoření výsledků byly delší než u srovnávaných algoritmů což se dalo očekávat, jelikož jsem se v rámci této práce měl zaměřit hlavně na vizuální kvalitu. Pro zlepšení rychlosti zpracování by bylo možné aplikaci rozšířit o vhodnější hierarchickou strukturu a samozřejmě také výpočet přesunout na nějakou paralelní platformu. Tato demonstrační aplikace pracuje aktuálně na CPU.

V dnešní moderní době je ve filmovém průmyslu kladen velký důraz na vizuální kvalitu výtvarných děl, kde jsou vlasy a srst často nejpodstatnější součástí většiny digitálně vytvořených postav. K tomuto je metoda sledování paprsků ideálním kandidátem už prakticky od jejího vzniku. Díky neustálým požadavkům na dokonalejší kvalitu je tato oblast poměrně častým předmětem výzkumu a přesto ještě stále nebyl nalezen ucelený přístup, o kterém by se dalo říci, že je nejvhodnější co se týká výstupné obrazové kvality, tak i do rychlosti zpracování. To mě mimo jiné donutilo zamyslet se nad tímto tématem a navíc mě to také inspirovalo k vytvoření diplomové práce právě na toto téma.

8 Literatura

[1] Apodaca, A. A.; Gritz, L.: *Advanced RenderMan: Creating CGI for Motion Picture*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., první vydání, 1999, ISBN 1558606181.

[2] Cook, R. L.; Carpenter, L.; Catmull, E.: The Reyes Image Rendering Architecture. *SIGGRAPH Comput. Graph.*, ročník 21, č. 4, Srpen 1987: s. 95–102, ISSN 0097-8930, doi:10.1145/37402.37414.
URL http://doi.acm.org/10.1145/37402.37414

[3] Cook, R. L.; Carpenter, L.; Catmull, E.: The Reyes Image Rendering Architecture. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, New York, NY, USA: ACM, 1987, ISBN 0-89791-227-6, s. 95–102, doi:10.1145/37401.37414.
URL http://doi.acm.org/10.1145/37401.37414

[4] Cook, R. L.; Halstead, J.; Planck, M.; aj.: Stochastic Simplification of Aggregate Detail. *ACM Trans. Graph.*, ročník 26, č. 3, Červenec 2007, ISSN 0730-0301, doi:10.1145/1276377.1276476.
URL http://doi.acm.org/10.1145/1276377.1276476

[5] Cook, R. L.; Halstead, J.; Planck, M.; aj.: Stochastic Simplification of Aggregate Detail. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, New York, NY, USA: ACM, 2007, doi:10.1145/1275808.1276476.
URL http://doi.acm.org/10.1145/1275808.1276476

[6] Hou, Q.; Qin, H.; Li, W.; aj.: Micropolygon Ray Tracing with Defocus and Motion Blur. *ACM Trans. Graph.*, ročník 29, č. 4, Červenec 2010: s. 64:1–64:10, ISSN 0730-0301, doi:10.1145/1778765.1778801.
URL http://doi.acm.org/10.1145/1778765.1778801

[7] Hou, Q.; Qin, H.; Li, W.; aj.: Micropolygon Ray Tracing with Defocus and Motion Blur. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, New York, NY, USA: ACM, 2010, ISBN 978-1-4503-0210-4, s. 64:1–64:10, doi:10.1145/1833349.1778801.
URL http://doi.acm.org/10.1145/1833349.1778801

[8] Igehy, H.: Tracing Ray Differentials. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, ISBN 0-201-48560-5, s. 179–186, doi:10.1145/311535.311555.
URL http://dx.doi.org/10.1145/311535.311555

[9] Kajiya, J. T.; Kay, T. L.: Rendering Fur with Three Dimensional Textures. *SIGGRAPH Comput. Graph.*, ročník 23, č. 3, Červenec 1989: s. 271–280, ISSN 0097-8930, doi:10.1145/74334.74361.
URL http://doi.acm.org/10.1145/74334.74361

[10] Kajiya, J. T.; Kay, T. L.: Rendering Fur with Three Dimensional Textures. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '89, New York, NY, USA: ACM, 1989, ISBN 0-89791-312-4, s. 271–280, doi:10.1145/74333.74361.
URL http://doi.acm.org/10.1145/74333.74361

[11] Marschner, S. R.; Jensen, H. W.; Cammarano, M.; aj.: Light Scattering from Human Hair Fibers. *ACM Trans. Graph.*, ročník 22, č. 3, Červenec 2003: s. 780–791, ISSN 0730-0301, doi:10.1145/882262.882345.
URL http://doi.acm.org/10.1145/882262.882345

[12] Marschner, S. R.; Jensen, H. W.; Cammarano, M.; aj.: Light Scattering from Human Hair Fibers. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, New York, NY, USA: ACM, 2003, ISBN 1-58113-709-5, s. 780–791, doi:10.1145/1201775.882345.
URL http://doi.acm.org/10.1145/1201775.882345

[13] Nakamaru, K.; Matsuoka, T.; Fujita, M.: Distance Aware Ray Tracing for Curves. In *ACM SIGGRAPH 2012 Posters*, SIGGRAPH '12, New York, NY, USA: ACM, 2012, ISBN 978-1-4503-1682-8, s. 103:1–103:1, doi:10.1145/2342896.2343016.
URL http://doi.acm.org/10.1145/2342896.2343016

[14] Nakamaru, K.; Ohno, Y.: Ray tracing for curves primitive. In *Journal of WSCG*, WSCG '2002, 2002, ISSN 1213-6980, s. 311–316.

[15] Neulander, I.: Fast Furry Ray Gathering. In *ACM SIGGRAPH 2010 Talks*, SIGGRAPH '10, New York, NY, USA: ACM, 2010, ISBN 978-1-4503-0394-1, s. 2:1–2:1, doi:10.1145/1837026.1837029.
URL http://doi.acm.org/10.1145/1837026.1837029

[16] Phong, B. T.: Illumination for Computer Generated Pictures. *Commun. ACM*, ročník 18, č. 6, Červen 1975: s. 311–317, ISSN 0001-0782, doi:10.1145/360825.360839.
URL http://doi.acm.org/10.1145/360825.360839

[17] Piegl, L.; Tiller, W.: *The NURBS Book (2Nd Ed.)*. New York, NY, USA: Springer-Verlag New York, Inc., 1997, ISBN 3-540-61545-8.

[18] Qin, H.; Chai, M.; Hou, Q.; aj.: Cone Tracing for Furry Object Rendering. *Visualization and Computer Graphics, IEEE Transactions on*, ročník 20, č. 8, Aug 2014: s. 1178–1188, ISSN 1077-2626, doi:10.1109/TVCG.2013.270.

[19] Ren, Z.; Zhou, K.; Li, T.; aj.: Interactive Hair Rendering Under Environment Lighting. *ACM Trans. Graph.*, ročník 29, č. 4, Červenec 2010: s. 55:1–55:8, ISSN 0730-0301, doi:10.1145/1778765.1778792.
URL http://doi.acm.org/10.1145/1778765.1778792

[20] Ren, Z.; Zhou, K.; Li, T.; aj.: Interactive Hair Rendering Under Environment Lighting. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, New York, NY, USA:

ACM, 2010, ISBN 978-1-4503-0210-4, s. 55:1–55:8, doi:10.1145/1833349.1778792.
URL http://doi.acm.org/10.1145/1833349.1778792

[21] Ryu, D.: 500 Million and Counting: Hair Rendering on Ratatouille. In *ACM SIGGRAPH 2007 Sketches*, SIGGRAPH '07, New York, NY, USA: ACM, 2007, doi:10.1145/1278780.1278842.
URL http://doi.acm.org/10.1145/1278780.1278842

[22] Struik, D.: *Lectures on Classical Differential Geometry*. Addison-Wesley series in mathematics, Addison-Wesley Publishing Company, 1961, ISBN 9780486656090.
URL http://books.google.cz/books?id=TIXg4hSXmFAC

[23] van Swaaij, M.: Ray-tracing Fur for Ice Age: The Melt Down. In *ACM SIGGRAPH 2006 Sketches*, SIGGRAPH '06, New York, NY, USA: ACM, 2006, ISBN 1-59593-364-6, doi:10.1145/1179849.1179904.
URL http://doi.acm.org/10.1145/1179849.1179904

[24] Wald, I.; Ize, T.; Kensler, A.; aj.: Ray Tracing Animated Scenes Using Coherent Grid Traversal. *ACM Trans. Graph.*, ročník 25, č. 3, Červenec 2006: s. 485–493, ISSN 0730-0301, doi:10.1145/1141911.1141913.
URL http://doi.acm.org/10.1145/1141911.1141913

[25] Wald, I.; Ize, T.; Kensler, A.; aj.: Ray Tracing Animated Scenes Using Coherent Grid Traversal. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, New York, NY, USA: ACM, 2006, ISBN 1-59593-364-6, s. 485–493, doi:10.1145/1179352.1141913.
URL http://doi.acm.org/10.1145/1179352.1141913

[26] Wang, G.: The subdivision method for finding the intersection between two bézier curves or surfaces. *Zhejiang University Journal*.

[27] Williams, A.; Barrus, S.; Morley, R. K.; aj.: An Efficient and Robust Ray-Box Intersection Algorithm. *J. Graphics Tools*, ročník 10, č. 1, 2005: s. 49–54.

[28] Zinke, A.; Weber, A.: Light Scattering from Filaments. *IEEE Transactions on Visualization and Computer Graphics*, ročník 13, č. 2, Březen 2007: s. 342–356, ISSN 1077-2626, doi:10.1109/TVCG.2007.43.
URL http://dx.doi.org/10.1109/TVCG.2007.43

A Obsah CD

K mé práci je přiloženo jedno CD, které obsahuje zdrojové kódy demonstrační aplikace vytvořené v rámci této diplomové práce. Dále je zde digitální verze této písemné zprávy a její zdrojový tvar. CD obsahuje následující složky.

- **Písemná zpráva** Složka obsahuje tento dokument ve formátu PDF a jeho zdrojový tvar v jazyce LaTeX.
- **Demonstrační aplikace** Složka obsahuje zdrojové kódy demonstrační aplikace, návod k její instalaci a ovládání. Složka dále obsahuje dokumentaci vytvořenou ze zdrojového kódu pomocí nástroje Doxygen a binární tvar aplikace přeložené pro 64-bitovou verzi operačního systémem Windows 7.